COMBAT IDENTIFICATION WITH SEQUENTIAL OBSERVATIONS, REJECTION OPTION, AND OUT-OF-LIBRARY TARGETS

DISSERTATION

Timothy W. Albrecht, Major, USAF

AFIT/DS/ENS/05-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense or the United States Government.

AFIT/DS/ENS/05-03

COMBAT IDENTIFICATION WITH SEQUENTIAL OBSERVATIONS, REJECTION OPTION, AND OUT-OF-LIBRARY TARGETS

DISSERTATION

Presented to the Faculty
Department of Operational Sciences
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy in Operations Research

Timothy W. Albrecht, B.S., M.S.
Major, USAF

September 2005

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT/DS/ENS/05-03

COMBAT IDENTIFICATION WITH SEQUENTIAL OBSERVATIONS, REJECTION OPTION, AND OUT-OF-LIBRARY TARGETS

Timothy W. Albrecht, B.S., M.S.
Major, USAF

Approved:

//Signed//	6 Sep 2005
Dr. Kenneth W. Bauer Jr. Committee Chairman	Date
//Signed//	6 Sep 2005
Dr. Mark E. Oxley Committee Member	Date
//Signed//	6 Sep 2005
Dr. John O. Miller Committee Member	Date
//Signed//	7 Sep 2005
Dr. Steven C. Gustafson Dean's Representative	Date

Accepted:

//Signed//	8 Sep 2005
Dr. Robert A. Calico Jr. Dean, Graduate School of Engineering and Management	Date

Abstract

Combat target identification (CID) is the process by which detected objects are characterized pursuant to military action. Errors in CID such as mis-labeling targets and non-targets carry significant costs. Fusing data from multiple sources and allowing a rejection, or non-declare, option can improve CID error rates.

This research extends a mathematical framework that selects the optimal sensor ensemble and fusion method across multiple decision thresholds subject to warfighter constraints. The formulation includes treatment of exemplars from target classes on which the CID system classifiers are not trained (out-of-library classes), and it enables the warfighter to optimize a CID system without explicit enumeration of classifier error costs.

A time-series classifier design methodology is developed and applied, resulting in a multi-variate Gaussian hidden Markov model (HMM) with a specially constructed hidden state space. The extended CID framework is used to compete the HMM-based CID system against a template-based CID system. The assessment uses a real world synthetic aperture radar (SAR) data collection comprised of ten in-library target classes and five out-of-library target classes. The framework evaluates competing classifier systems that use multiple fusion methods, including neural network fusion and label fusion, varied prior probabilities of targets and non-targets, varied correlation between multiple sensor looks, and varied levels of target pose estimation error. Also, an on-line target pose estimator is developed using principal component analysis of masked target SAR images. This estimator validates experimental assumptions on target pose prior to classification.

The CID system assessment using the extended framework reveals larger feasible operating regions for the HMM-based classifier across experimental settings. In some cases the HMM-based classifier yields a feasible region that is 25% of the

threshold operating space versus 1% for the template-based classifier. Similar performance results are obtained for rule-based label fusion and the more complex neural network fusion and are explained by the new ability to independently set classifier thresholds with the label fusion method.

Acknowledgements

Dissertations are not researched and written by the student in a vacuum. It is fitting to recognize for posterity those people who collaboratively influenced, assisted, molded, and refined this research into the final product printed here.

The most important decision a student makes is the choice of research advisor. Dr. Ken Bauer combines the much sought after qualities of experience, energy, and vision. I am grateful he took me on as his student and guided me through the program. I am also grateful to my research committee, Drs. Mark Oxley, J. O. Miller, and Steven Gustafson, for lending their time, support, and critical eye throughout this research effort.

I would also like to thank the sponsors of this research, AFOSR, ACC/DR and AFRL/SN, and recognize the significant role Mr. Chuck Sadowski played in framing the CID problem.

Construction forced AFIT's PhD students into temporary office space; the infamous BARF trailer. The office move coincided with an initiative to get AF members back into shape. Both actions were fortuitous for the 2005 PhD class: not only did we become pretty good ultimate frisbee players, but we also benefitted from the synergistic effect of belonging to a close-knit multi-disciplined research group.

I'd like to express my gratitude to those ENS PhD students who went before me and whose sage advice helped in innumerable ways. Lance, Steve, Todd, and especially Trevor, thank you for easing the pain of coursework, research, and writing.

And finally, I owe a special "*thank you*" to my wife for her patience, love, and support, and to our sons whose lives at the other end of my life's spectrum provide wonderful distraction and relief from the absorbing state that is dissertation research.

Timothy W. Albrecht

Table of Contents

List of Figures

List of Tables

Combat Identification with Sequential Observations, Rejection Option, and Out-of-Library Targets

1. Introduction

1.1 Background

The research reported in this dissertation stems from a study of pattern recognition applied to modern warfare. Two thousand years ago the Chinese military philosopher Sun Tzu wrote, "if you know the enemy and know yourself, you need not fear the results of one hundred battles [1]." Thus, perfect knowledge of your enemy, his assets and their location coupled with knowledge of your own assets, locations, and capabilities provide the military leader an undeniable advantage over his adversary.

United States Armed Forces doctrine, and US Air Force (USAF) doctrine in particular, have made the ancient truism the official practice of the US military. USAF doctrine document AFDD 2-1, entitled *Air Warfare*, relates that if an enemy's key targets can be found and identified, then air power can be applied [2]. Thus, identifying, or classifying, a target is a critical link in the kill chain that begins with finding a target, includes engaging the target, and ends with assessing the outcome of the engagement. The US military defines combat identification (CID) as

> the process of attaining an accurate characterization of detected objects in the joint battlespace to the extent that high confidence, timely application of military options and weapons resources can occur [3].

Figure 1 depicts the CID problem from the combat pilot's perspective. The true nature of the entities sharing the battlespace is unknown. Here CID characterizes those entities using information from a variety of sources. The goal of CID is to maximize operational effectiveness by neutralizing the enemy with an efficient allocation

Figure 1. The real combat identification problem: battlespace characterization from the combat pilot's perspective. Figure originally presented by Mr. Charles Sadowski, ACC/DRSA [4].

of combat resources while minimizing friendly casualties [4]. Friendly casualties may result from either enemy or friendly fire, commonly called fratricide. By improving CID performance, friendly casualties are reduced on both fronts: fewer enemy to engage friendly units, and fewer mis-identified friendly units.

Doctrinal links with CID can be found in joint and Air Force doctrine. In *Joint Vision 2020* the Chairman of the Joint Chiefs of Staff provides a template for the transformation of the US Armed Forces. In this document CID impacts three of four operational concepts: precision engagement, dominant maneuver, and full dimensional protection [5].

The US Armed Forces recognize the principles of war as fundamental guidance for the application of military power. They are listed and defined in *Joint Warfare of the Armed Forces of the United States*, JP 1, the capstone joint warfare doctrine document [6]. Accurately identifying targets in a timely manner supports the principles of offense, economy of force, and surprise, thus affording an advantage over an adversary without a similar capability.

Among the seven tenets of aerospace power which complement the principles of war and reflect the evolution of airpower, Air Force doctrine lists decentralized execution of air and space power. Decentralized execution is

> the delegation of execution authority to responsible and capable lower-level commanders to achieve effective span of control and to foster disciplined initiative, situational responsiveness, and tactical flexibility. It allows subordinates to exploit opportunities in rapidly changing, fluid situations. [7]

Improved target recognition systems allow operators to respond quickly and provide greater flexibility in their responsiveness.

Air Force Basic Doctrine, AFDD 1, lists six distinctive capabilities, or areas of expertise, of the Air Force. Of these six distinctive capabilities, *Global Attack* and *Precision Engagement* are directly impacted by improvements to target recognition systems. *Global Attack* refers to the "ability of the Air Force to attack rapidly and persistently with a wide range of munitions anywhere on the globe at any time [7]." *Precision Engagement* refers to air and space power's ability "to apply discriminate force precisely where required [7]."

At a more detailed level of airpower application, AFDD 1 lists seventeen key operational functions of the Air Force. Of those listed, improved target recognition systems positively impact the following functions:

- *Strategic Attack*, defined as offensive action conducted by command authorities aimed at generating effects that most directly achieve national security objectives by affecting the adversary's leadership, conflict-sustaining resources, and strategy
- *Counterair*, defined as operations that attain and maintain a desired degree of air superiority by the destruction, degradation, or disruption of enemy forces
- *Counterland*, defined as air and space operations against enemy land force capabilities to create effects that achieve JFC (Joint Forces Commander) objectives
- *Countersea*, defined as functions that extend Air Force capabilities into a maritime environment

- *Surveillance and Reconnaissance*, defined as systematically observing air, space, surface, or subsurface areas, places, persons, or things, by visual, aural, electronic, photographic, or other means ... designed to provide warning of enemy initiatives and threats and to detect changes in enemy activities [7]

The last function listed above, Surveillance and Reconnaissance, is covered more fully in two top-level Air Force doctrine documents: AFDD 2-5.2, *Intelligence, Surveillance, and Reconnaissance Operations* [8], and AFPAM 14-210, *United States Air Force Targeting Guide* [9], where AFDD 2-5.2 outlines the principles and doctrine for intelligence, surveillance, and reconnaissance (ISR), and AFPAM 14-210 explains the principles and concepts of targeting, a core Air Force discipline which integrates intelligence information about targets with operational information about friendly objectives, capabilities, and doctrine.

Both documents describe the process of information fusion. The ISR-derived information from many sources is combined, evaluated, and analyzed in a process called fusion. Fusion is listed as one of eleven ISR principles in AFDD 2-5.2 [8], and AFPAM 14-210 defines fusion as the process of combining multi-source data into intelligence necessary for decision making and highlights fusion as a guiding principle in the targeting process.

While identifying and defining fusion as an important principle in intelligence gathering and processing, neither document provides guidance for carrying out multi-source fusion. Indeed, intelligently automating the fusion of information from multiple sources, or sensors, would improve ISR operations by making more accurate target identifications and would speed the targeting timeline by lessening reliance on human interpretation.

The Air Force places great emphasis on the importance of recognizing and fostering technological advances in order to improve warfighting capabilities. The AFDD 1 describes *Technology-to-warfighting*, one of three Air Force core competencies, as follows:

> As a leader in the military application of air, space, and intelligence, surveillance, and reconnaissance technology, the Air Force is committed to innovation to guide research, development, and fielding of unsurpassed capabilities. Just as the advent of powered flight revolutionized joint warfighting, recent advances in low observable technologies; space-based systems; manipulation of information; precision; and small, smart weapons offer no less dramatic advantages for combatant commanders. The Air Force nurtures and promotes its ability to translate our technology into operational capability to prevail in conflict and avert technological surprise. [7]

Research in the area of target recognition systems fits directly under the umbrella of this core competency of the Air Force and is supported by Joint and Air Force doctrine.

1.2 Problem Statement

With sound doctrinal support for research in the area of CID explained in Section 1.1, this section details problems addressed by this dissertation. A notional CID system is shown in Figure 2. Observations through time of a region of interest are made by two sensors, s_1 and s_2. Sensor data D is processed into features F which are then classified into labels L before being fused into final labels L_{final}.

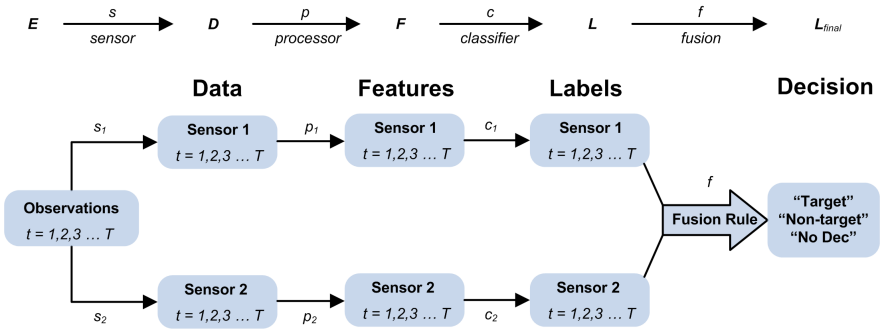

Figure 2. Notional CID system with two sensors evaluating observations through time $t = T$.

Air Force doctrine stipulates that the targeting process must gather information to reach a desired level of labeling confidence prior to making a shoot deci-

sion [8, 2]. Two paths to improved classifier confidence are temporal fusion, or fusion of sequential observations, and sensor fusion, or fusion across sensors. Both fusion methods attempt to improve classification performance by combining information contained in multiple observations. With temporal fusion the classification system processes a sequence of event observations. Observations may be autocorrelated and additional observations may provide information beneficial to the classification process, or they may confuse the classifier, producing undesirable results.

Fusion of multiple sensors is considered when designing multiple classifier systems (MCS). The architect must design both an ensemble of classifiers and a fusion rule with which to combine the individual classifier outputs. The MCS performance depends on an ensemble whose classifiers make disjoint errors (i.e., classifier A and classifier B errors occur in non-overlapping areas of the feature space), and a fusion rule which takes advantage of relative strengths of the constituent classifiers [10].

Given a CID system, the warfighter requires a label-space that is less rigid than forced-decision [4]. A forced-decision classifier trained to recognize objects in class A, B, C, or D maps every test record into one of four possible classes. Warfighters require that a reject option be given to the classifier which allows it to opt against the forced-decision label and for a "*non-declaration*" label.

Thus, the warfighter requires at least a trichotomous label space for the CID system. Using the example above, data class A is labeled "*hostile*", data classes B, C, and D are labeled "*friend*", and when the classifier does not achieve the desired labeling confidence it applies the third label, "*non-declared*".

Optimizing classifiers with a reject option has been studied [11, 12, 13, 14], but invariably the optimal decision boundaries rely on a set cost rule for classifier errors. Laine's research [15] proposes a methodology for optimizing a rejection-capable CID system without explicit error costs. Thus the warfighter does not specify the relative cost of a fratricide incident versus collateral damage versus a successful engagement.

One useful extension of the trichotomous label-space of a rejection-capable CID system is the incorporation of an "*out-of-library*" label. A CID system can be thought of as a simple classifier trained on exemplars from a specified set of target classes. The union of the target classes constitutes the library of the classifier. An exemplar is said to be "*in-library*" if it is from a target class which the classifier has been trained to recognize, and it is "*out-of-library*" otherwise. It is likely that a fielded CID system will encounter targets in out-of-library classes [4].

The goals of this research include the development of a robust, time-series MCS for use in an extended CID optimization framework that includes both a rejection option and in-library and out-of-library discrimination. In addition, the effects of data correlation in a temporally-fused MCS, data prevalence, and extended operating conditions are examined. Also, means of performance assessment are developed.

For the foreseeable future air operations will require timely acquisition of and precise engagement against targets regardless of environmental conditions while minimizing collateral damage. This research focuses on the sensor processing and decision making parts of the kill chain.

1.3 Scope

The scope of research for this dissertation includes the methodology used to design a temporally-fused MCS in a CID setting. Much attention in the Department of Defense has been paid to the development of a family-of-systems (FOS) networked together to provide a joint service CID capability [16]. Some of these FOS systems are cooperative identifiers, such as transponders which identify friendly forces by producing a certain signal. This research considers non-cooperative means of target classification.

Specifically, the research presented here uses synthetic aperture radar (SAR) imagery of ground targets collected from an airborne sensor. The imagery has been

pre-processed to present the researcher with a detected target in each image. Thus, the focus is not target detection, but rather target classification.

This research advances the field of pattern recognition by developing a temporally-fused, multiple classifier CID system using hidden Markov models (HMMs) operating on features drawn from SAR images of ground targets taken at various aspect angles. Sequencing the target observations by aspect angle generates a temporal sensor-target relationship. A real world application is classification of ground targets by an airborne sensor in a multi-look, or sequence of observations, setting with an unknown relative initial aspect angle of target to sensor.

This research also explores the impact of extended operating conditions (EOCs) on CID systems. Sensor observation of a specific ground target presents different signatures depending on the sensor-target orientation, target class variant, target articulation, and the surrounding clutter environment. A ground vehicle has different signatures when observed head-on versus from its flank. Similarly, a turreted target has different signatures if its barrel is in-line with the body versus rotated askew the body. The EOCs are real-world considerations which degrade classifier performance due to variations in the target. Synthesized data typically adds white noise to the signature that masks the target, while EOCs present targets whose signatures vary from the in-library exemplars.

1.4 Approach

The first step of the research process is a review of the pertinent literature and is presented in Chapter 2. Four areas are reviewed:

- Hidden Markov models as time series classifiers and their theory and application
- High range-resolution radar signature processing and its use in target recognition

- Model selection as discussed in information theory literature
- Multiple classifier systems, sensor fusion, and other CID issues

The review highlights current research efforts, develops supporting theory, and points to areas that contribute to this dissertation research.

Next, the temporally-fused CID system is designed using a heuristic methodology to specify the model. The methodology focuses on classifier performance while selecting model complexity and structure. In addition, the CID system incorporates an in-library versus out-of-library discriminator designed using a separate heuristic methodology focused on two-class separability. The out-of-library discriminator is used to extend Laine's CID optimization framework [15] by including both an "*out-of-library*" label and the associated warfighter constraint used in optimizing the CID system.

By exploring separate design methodologies, the research finds robust architectures that perform well in an EOC setting, where the in-library target data is significantly different from the in-library training data and may include out-of-library exemplars.

1.5 Contributions

Contributions from this dissertation research are in the following areas:

- Development of an HMM-based time series classifier
- Extension of Laine's CID optimization framework to include out-of-library performance
- Development of an out-of-library classification methodology
- Development of a target pose-estimation methodology using principal component analysis

- Application of the extended framework to a multi-class ATR experiment that competes the HMM-based classifier against a template-based classifier
- Development of the framework to allow classifiers to make reject, or not declare, decisions, to test classifiers against out-of-library records, and to measure the performance of three different fusion methods
- Development of evidence for independent optimal threshold settings for label fusion

A comprehensive review of the literature covers the theory and development of hidden Markov models. The application of HMMs to ATR problems using high range-resolution radar signatures as features is described in Sec. 2.1.3.10, and it reveals limitations in treatment of prior knowledge of target aspect, inclusion of a rejection option, and performance considering out-of-library targets. Other research areas covered in the literature review include model complexity in HMMs, multiple classifier fusion, rejection theory, and Laine's CID optimization framework.

Chapter 3 describes the development of an HMM-based time series classifier. Ultimately, the methodology results in a multi-dimensional Gaussian HMM operating on HRR-derived feature data. The model takes as input a sequence of feature data ordered by target aspect angle. The model establishes a relation between the observation distribution associated with each hidden state and the signature of the target within a range of aspect angle.

Chapter 4 extends Laine's CID optimization framework by including an out-of-library performance measure. The framework retains the desired characteristic of allowing trade-off analysis without explicit classification error costs.

Section 4.3.2.5 describes a methodology whereby a classifier assigns an estimated posterior probability of out-of-library class membership to a test record. This methodology is implemented as a post-processing step after the classifier trained on in-library classes has adjudicated the test record. The methodology produces the

estimated out-of-library posterior probability as a function of the in-library class posterior probabilities produced by the classifier.

Section 5.4.5 develops a method to estimate target aspect angle based on a target mask of a SAR image. The method uses principal component analysis to find the major axis of the target mask. An initial experiment finds pose estimation error to be roughly 11°.

Chapter 5 details the application of the extended CID framework to an ATR experiment using DCS radar SAR data. The experiment competes an HMM-based system (a derivative of the Chapter 3 system) against a template-based classifier. The extended framework allows the systems to be compared inclusive of warfighter constraints, rejection option, and out-of-library target records. Results show that the HMM-based system provides the warfighter with better and more robust performance across a variety of experiment settings, including fusion rule, hostile/friend class prevalence, observation length, and prior knowledge of target aspect angle. Also, the size of feasible region in the threshold space provides a simple comparative measure of classifier robustness, and performance surfaces efficiently communicate performance information and trade-space.

Laine's research [15] has shown that independent thresholding for each classifier prior to applying a label fusion rule allows improved performance over the application of single thresholding after the fusion of classifier outputs. Section 5.6.2.5 shows that independent thresholding enables each classifier to use optimal thresholds in different locations in the threshold space. This added flexibility allows the label fusion method to combine a classifier whose threshold setting allows it to perform well in one performance measure, but poorly elsewhere, with a second classifier whose threshold setting allows it to perform well in another performance area.

1.6 Organization

The remainder of the document is organized as follows:

Chapter 2 provides background instruction, describes key supporting areas of the proposed research, and provides a review of the current literature. The supporting research areas include: hidden Markov models (their theory and application), high range-resolution radar profiles (their processing and use in automatic target recognition (ATR)), model selection and model complexity as discussed in information theory, and the design of multiple classifier systems (or sensor fusion).

Chapter 3 describes a heuristic approach to the design and development of an HMM-based classifier. First, an example application of an HMM-based classifier to sequences of genetic data is given. Next, model selection theory is applied in the choice of HMM design. Finally, a series of refinements to the HMM design are made with regard to assumptions and proven performance.

Chapter 4 describes the proposed classifier and CID framework extension, and the proposed HMM-based classifier is presented as part of an extended CID optimization framework which includes both a rejection option and out-of-library exemplars.

Chapter 5 considers application to DSC data and a competitor, where the proposed classifier is compared to a template-based classifier using SAR data from a 2004 collection within the extended CID framework.

Chapter 6 presents a summary of findings, discusses research contributions, and proposes future research areas stemming from this work.

2. Background

This chapter reviews pertinent literature, provides background information, and is organized by research area. First, hidden Markov models as time series classifiers are introduced, related literature is reviewed, and supporting theory is shown. Second, high range-resolution radar as a source of classification features is covered. Third, model selection in the context of information theory is defined and related literature is reviewed. Finally, basic concepts and taxonomies of sensor fusion and multiple classifier systems are covered.

2.1 Hidden Markov models

2.1.1 Introduction

An important aspect of combat identification (CID) is the incorporation of temporal target observations into the classification process. In his dissertation Fielding proves that given a sequence of observations in which there is a provable dependency, the entropy of the joint observations is less than the entropy of the individual observations [17]. Thus, a classifier operating on the greater source of information (less entropy) will have equal or greater classification power than single-look methods.

A review of the pattern recognition literature in search of time series classifiers, or classifiers which incorporate data order, yields hidden Markov models (HMMs) as the primary classifier for the research presented here. In the following sections hidden Markov models are introduced, their mathematical development is given, and HMM applications in the field of automatic target recognition are reviewed.

2.1.2 Literature

Hidden Markov models fit into a broad class of statistical signal models which also includes Gaussian processes, Poisson processes and Markov processes. These

models seek to characterize a signal as a parametric random process whose parameters can be estimated (or determined) in a well-defined manner [18]. HMMs are statistical representations of time series data. An HMM is used to represent probability distributions given a sequence, or many sequences, of observations. The fundamental property of HMMs is the assumption that the sequence of observations is a noisy function of a Markov chain which is not directly observed, or hidden.

The literature contains several HMM tutorial articles and texts. Rabiner's tutorial on HMMs [18] is widely cited and gives an introduction to HMMs applies them in a speech recognition application. A more recent article on HMMs and their development is in Ghahramani's paper [19]. A small section introducing HMMs in Duda and Hart's classic pattern recognition text [20] is useful for its diagrams.

Two texts wholly dedicated to HMMs are useful in researching the variety of specialized HMMs and their applications. Elliott's text [21] focuses on signal processing applications of HMMs, and MacDonald's text [22] focuses on discrete-valued time series applications. Both provide excellent mathematical development for HMMs.

A history of HMMs begins with their introduction as probabilistic functions of Markov chains in Baum and Petrie's 1966 paper [23]. Later, Baum, Petrie, Soules, and Weiss introduced a method to calculate the conditional probability of a state given a sequence of observations [24]. In the same paper, they showed how to efficiently estimate the parameters of an HMM. The algorithm, called alternately the Baum algorithm, the Baum-Petrie algorithm, or the Baum-Welch algorithm, is the expectation maximization algorithm of Dempster, Laird, and Rubin [25] applied to HMMs. Local convergence of the algorithm was proved [24], and later work [26, 27] proved the consistency and asymptotic normality of the maximum likelihood estimators of the HMM parameters.

Ephraim and Merhav provide a well-referenced overview of HMMs and their applications [28]. HMMs have been applied in a number of research areas. State

of the art speech recognition engines employ HMMs [18, 29, 30, 31] to match spoken word with stored language. The vast amounts of data generated in efforts to map genetic material are sorted by structure and purpose in an area of study called computational biology, and HMMs play a major role in the effort [32, 33, 34]. Examples of pattern recognition applications of HMMs are found in Arica [35] and Cai [36] where HMMs are used for character recognition, Hu [37] where HMMs are employed to classify facial emotions, and Krishnamurthy [38] where HMMs process signal information in the presence of noise. Gader [39] applies HMMs with ground penetrating radar to classify mine types.

2.1.3 Theory

This section draws from several sources in developing HMM notation, parameterization, and mathematical development. Rabiner [18] and Ghahramani [19] provide outstanding tutorials on HMMs and their applications. Bilmes [40] and Elliott [21] provide helpful development of HMM algorithms and their convergence theory.

2.1.3.1 Definition and notation

The notation follows the stochastic literature, specifically Kulkarni's stochastic system analysis text [41], and a blend of HMM notation as found in Elliott's text [21] and Rabiner's HMM tutorial [18].

In developing the theory of HMMs we begin with a stochastic process $\{X_n, n \geq 0\}$, where X_n denotes the state of the system at time n and where for all $n \geq 0$, X_n is a random variable taking values in set S. We further assume $\{X_n, n \geq 0\}$ is a discrete-time Markov chain (DTMC) with finite state space S such that

1. for all $n \geq 0, X_n \in S$ with probability 1,

2. $P\{X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0\} = P\{X_{n+1} = j | X_n = i\}$ for all $n \geq 0$ and $i, j \in S$, which is the first order Markov property.

Next we introduce the idea of time-homogeneity. A DTMC is time-homogeneous when the conditional probabilities $P\{X_{n+1} = j | X_n = i\}$ are independent of time, $n \geq 0$, for all $i, j \in S$.

In an HMM the finite-state space, time-homogeneous DTMC $\{X_n, n \geq 0\}$ is hidden and can only be observed through an additional stochastic process, $\{Y_n, n \geq 0\}$, which is a sequence of conditionally independent random variables with the conditional distribution of Y_n depending on the hidden DTMC $\{X_n, n \geq 0\}$ only through the state at time n, X_n. In a discrete HMM the state space of Y_n is finite.

Thus the discrete-time stochastic process $\{(X_n, Y_n), n \geq 0\}$ forms a hidden Markov model where $\{X_n, n \geq 0\}$ is a hidden time-homogeneous DTMC with finite state space and $\{Y_n, n \geq 0\}$ is an observation sequence dependent on the state of the hidden DTMC at time n.

2.1.3.2 Parameterization

To parameterize a discrete HMM a stochastic transition matrix, $A = [a_{ij}]$, where $a_{ij} = P\{X_{n+1} = j | X_n = i\}$, is used to represent the hidden Markov chain. In addition to the hidden state transition matrix, A, an observation distribution matrix, B, is defined by $[b_{ji}] = P\{Y_n = i | X_n = j\}$ where each observation Y_n of the sequence $\{Y_n, n \geq 0\}$ is one of Q possible values. Finally, the parameterization of a discrete HMM must include an initial state distribution, $\pi_i = P\{X_0 = i\}$, which provides the starting point for the hidden Markov chain. Thus, a discrete HMM with S hidden states and Q observation states is parameterized by a hidden state transition matrix, $A = [a_{ij}] \in \mathbb{R}^{S \times S}$, an observation distribution matrix, $B = [b_{ji}] \in \mathbb{R}^{S \times Q}$, and an initial state distribution, $\pi \in \mathbb{R}^S$. The complete set of parameters for a given

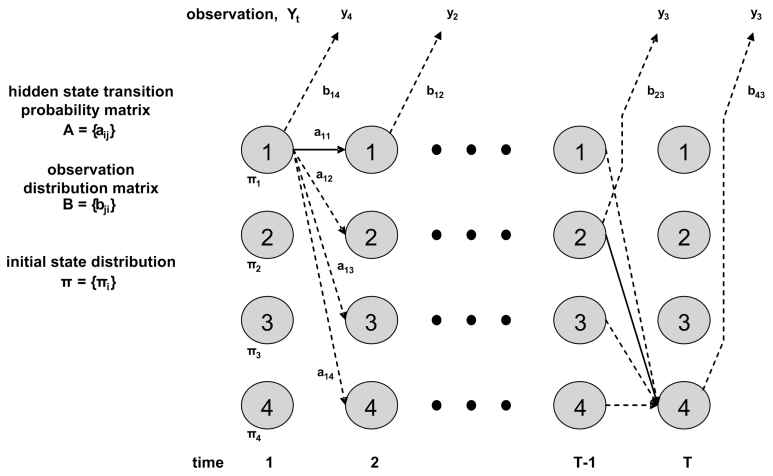

Figure 3. Trellis diagram of a discrete HMM with 4 hidden states, 4 observation symbols $\{y_1, y_2, y_3, y_4\}$, and sequence length T. Here A is the hidden state transition matrix and governs the progression of the hidden Markov chain, B is the observation distribution matrix and governs the sequence of observed symbols, and π is the initial state distribution and governs the starting state of the hidden Markov chain.

model is written, $\lambda = (A, B, \pi)$. Figure 3 provides a schematic representation of hidden Markov model parameterization and functioning.

2.1.3.3 Basic HMM problems

Rabiner discusses three basic problems associated with HMMs. The following derivations with slight modification to the notation can be found in his tutorial [18]:

- ***Evaluation*** Given a model λ evaluate the probability $P\{Y|\lambda\}$ of producing a specified observation sequence of length T, $Y \in \{Y_n\}_1^T$. Note that we have adjusted the time index to begin at $n = 1$ so that a sequence from time $n = 1$ to $n = T$ is of length T and not $T + 1$.
- ***Decoding*** Given a model λ and an observations sequence $Y \in \{Y_n\}_1^T$, find the best state sequence $X \in \{X_n\}_1^T$ that explains Y.

- ***Learning*** Given an observation sequence, or set of observation sequences, $\{Y\}$, parameterize an HMM, $\lambda^* = (A, B, \pi)$, such that it is the most likely model to have produced the given data. This amounts to training an HMM given observation data.

2.1.3.4 Evaluation Problem

The first HMM problem seeks to find the probability of producing an observation sequence of length T, $Y \in \{Y_n\}_1^T$, given a model, $\lambda = (A, B, \pi)$. The probability of producing an observation sequence $Y = Y_1 Y_2 \dots Y_T$ given a hidden state sequence $X \in \{X_n\}_1^T$ with $X = X_1 X_2 \dots X_T$ and the model, $\lambda = (A, B, \pi)$ is

$$P\{Y|X, \lambda\} = \prod_{n=1}^{T} P\{Y_n|X_n, \lambda\} = \prod_{n=1}^{T} b_{X_n, Y_n}, \tag{1}$$

where $b_{ji} \in B$, the observation distribution matrix. Thus $P\{Y|X, \lambda\} = b_{X_1 Y_1} b_{X_2 Y_2} \dots b_{X_T Y_T}$ which can be joined with the probability of a hidden state sequence given a model, $P\{X|\lambda\} = \pi_{X_1} a_{X_1 X_2} a_{X_2 X_3} \dots a_{X_{T-1} X_T}$ to yield the joint probability of an observation sequence and a hidden state sequence given a model

$$P\{Y, X|\lambda\} = P\{Y|X, \lambda\} \cdot P\{X|\lambda\}. \tag{2}$$

By summing over all hidden state sequences, the exact formulation for $P\{Y|\lambda\}$ is

$$\begin{aligned} P\{Y|\lambda\} &= \sum_{\text{all } X \in \{X_n\}} P\{Y|X, \lambda\} \cdot P\{X|\lambda\} \\ &= \sum_{\text{all } X_1 X_2 \dots X_T \in \{X_n\}} \pi_{X_1} b_{X_1 Y_1} \cdot a_{X_1 X_2} b_{X_2 Y_2} \cdots a_{X_{T-1} X_T} b_{X_T Y_T}. \end{aligned} \tag{3}$$

While this process yields an exact solution, it requires an intractable number of calculations, $2T \cdot S^T$, where T is the sequence length and S is the number of hidden states in the Markov chain. A recursive algorithm, called the forward procedure,

reduces the number of necessary calculations to a manageable level and provides an efficient means of calculating $P\{Y|\lambda\}$.

The forward variable, $\alpha_i(n)$, is defined as the probability of observing a partial sequence to time n and being in hidden state i at time n given a model λ, or

$$\alpha_i(n) = P\{Y_1 Y_2 \dots Y_n, X_n = i|\lambda\}. \tag{4}$$

Each $\alpha_i(n), 1 \leq n \leq T$, can be defined recursively given an initial α_i.

1. Initialization step: $\alpha_i(1) = \pi_i b_{iY_1}$ which is the probability of starting in state i and observing Y_1.
2. Inductive step: $\alpha_j(n+1) = \left[\sum_{i=1}^{S} \alpha_i(n) a_{ij}\right] b_{jY_{n+1}}$ which is the probability of being in state j at time $n+1$ and observing the sequence $Y_1 Y_2 \dots Y_{n+1}$. The bracketed portion describes the probability of arriving at state j at time $n+1$ from state i at time n. By necessity the Markov chain must be in one of S states at time n. Summing $\alpha_i(n)$ over S states accounts for all possible one-step starting points at time n. Multiplying by $b_{jY_{n+1}}$ concludes the inductive step by incorporating the probability of being in state j at time $n+1$ and observing y_{n+1}.
3. Termination step: $P\{Y|\lambda\} = \sum_{i=1}^{S} \alpha_i(T)$ where T is the observation sequence length.

The recursive algorithm reduces the computational complexity of finding $P\{Y|\lambda\}$ from $2T \cdot S^T$ to $S^2 T$. Figure 4 uses the trellis schematic to illustrate calculation of the forward variable.

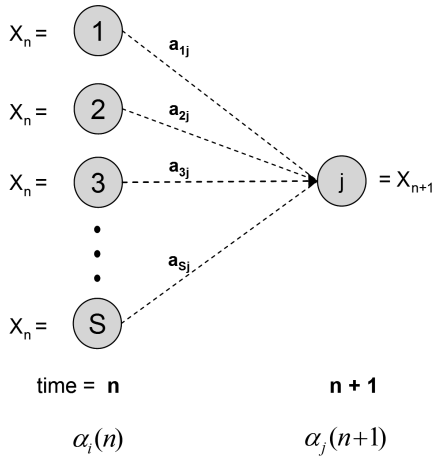

Figure 4. Diagram illustrating the hidden Markov chain propagation from time n to $n+1$ for the forward variable. The process begins in state i, one of S states, and transitions according to the state transition matrix, $A = [a_{ij}]$, to state j.

2.1.3.5 Decoding Problem

The backward variable is similar to the forward variable and plays a role in the solution to the second HMM problem: given a model λ and an observations sequence $Y \in \{Y_n\}_1^T$, find the state sequence $X \in \{X_n\}_1^T$ that best explains Y.

The backward variable $\beta_i(n)$ is defined as the probability of observing a partial sequence from time $n+1$ to time T given a model λ and that the model is in state i at time n, or

$$\beta_i(n) = P\{Y_{n+1}Y_{n+2}\ldots Y_T | X_n = i, \lambda\}. \tag{5}$$

Again, three steps are used efficiently calculate $P\{Y|\lambda\}$ using the newly defined backward variable (see Fig. 5):

1. Initialization step: $\beta_i(T) = 1$ which arbitrarily assigns a probability of 1 to each partial sequence.
2. Inductive step: $\beta_i(n) = \sum_{j=1}^{S} a_{ij} b_{jY_{n+1}} \beta_j(n+1)$ which is the probability of observing the partial sequence $Y_{n+1}Y_{n+2}\ldots Y_T$ given the model and that the

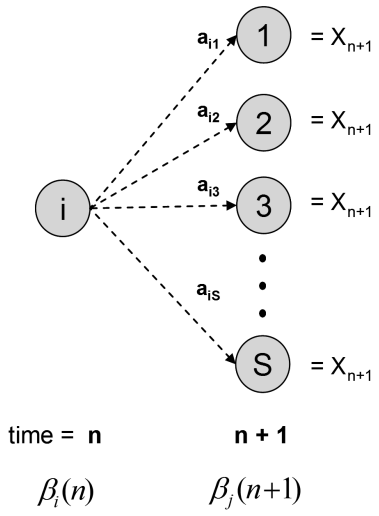

Figure 5. Diagram illustrating the hidden Markov chain propagation from time n to $n+1$ for the backward variable. The process begins in state i and transitions according to the state transition matrix, $A = [a_{ij}]$, to state j, one of S states.

model is in state i at time n. The product $a_{ij}b_{jY_{n+1}}$ gives the probability of making a single time-step transition from state i to state j and observing Y_{n+1}. Multiplying further by $\beta_j(n+1)$ incorporates the recursive element which accounts for the remaining sequence steps. Summing over the S possible states accounts for the S possible one-step end states from time n to time $n+1$.

3. Termination step: $P\{Y|\lambda\} = \sum_{i=1}^{S} \beta_i(1)\pi_i b_{iY_1}$

An additional variable, $\gamma_i(n)$, is needed to solve the second HMM problem, where $\gamma_i(n)$ is defined as the probability of being in state i at time n given the observation sequence $Y \in \{Y_n\}_1^T$ and the model λ, or

$$\gamma_i(n) = P\{X_n = i|Y, \lambda\}. \tag{6}$$

Note that,

$$P\{X_n = i|Y, \lambda\} = \frac{P\{Y, X_n = i|\lambda\}}{P\{Y|\lambda\}} = \frac{P\{Y, X_n = i|\lambda\}}{\sum_{j=1}^{S} P\{Y, X_n = j|\lambda\}}, \tag{7}$$

and with the following use of the forward and backward variables,

$$\alpha_i(n)\cdot\beta_i(n) = P\{Y_1Y_2\ldots Y_n, X_n = i|\lambda\}\cdot P\{Y_{n+1}Y_{n+2}\ldots Y_T|X_n = i,\lambda\} = P\{Y, X_n = i|\lambda\},$$

we can write

$$\gamma_i(n) = \frac{\alpha_i(n)\cdot\beta_i(n)}{\sum_{j=1}^{S}\alpha_j(n)\cdot\beta_j(n)}. \tag{8}$$

Then, to solve the second HMM problem and find the sequence of the individually most likely hidden states, $\{X_n^*\} = X_1X_2\ldots X_T$, we make the following comparison at each step of the sequence

$$X_n = i \quad \text{such that} \quad i = \underset{1\le i\le S}{\operatorname{argmax}}\left[\gamma_i(n)\right] \qquad \text{for } 1\le n\le T. \tag{9}$$

2.1.3.6 Learning Problem

The third and most complicated HMM problem seeks to update the model parameters, $\lambda = (A, B, \pi)$, to maximize the probability of the observation sequence given the model. The most commonly-used algorithm for this task is the Baum-Welch algorithm [23, 24]. Its derivation is shown here in two separate ways: first, following the notation used thus far, and second, in notation more appropriate to expectation maximization studies.

An additional variable, $\xi_{ij}(n)$, is needed to solve the third HMM problem (see Fig. 6). It is defined as the joint probability of being in state i at time n and being in state j at time $n+1$ given an observation sequence and a model, or

$$\xi_{ij}(n) = P\{X_n = i, X_{n+1} = j|Y,\lambda\}. \tag{10}$$

Expanding the definition using the given observation sequence gives

$$\xi_{ij}(n) = \frac{P\{X_n = i, X_{n+1} = j, Y|\lambda\}}{P\{Y|\lambda\}}. \tag{11}$$

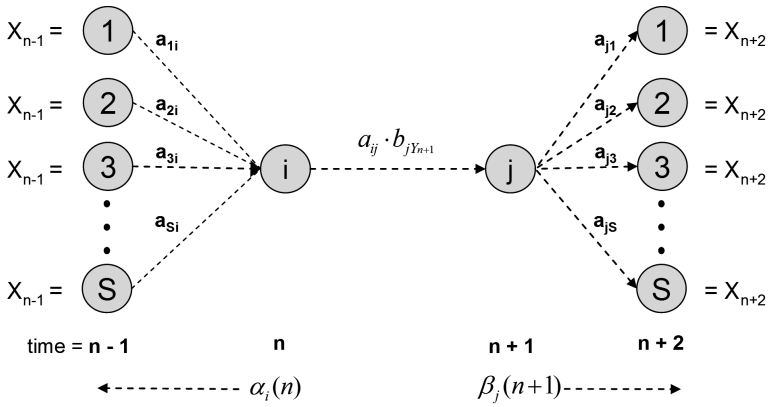

Figure 6. Diagram illustrating the hidden Markov chain propagation from time n to $n+1$ for $\xi_{ij}(n)$ incorporating the forward and backward variables.

Incorporating the forward and backward variables gives

$$\xi_{ij}(n) = \frac{\alpha_i(n) \cdot a_{ij} b_{jY_{n+1}} \cdot \beta_j(n+1)}{P\{Y|\lambda\}}. \tag{12}$$

Summing over all possible one-step state transitions yields

$$\xi_{ij}(n) = \frac{\alpha_i(n) \cdot a_{ij} b_{jY_{n+1}} \cdot \beta_j(n+1)}{\sum_{i=1}^{S} \sum_{j=1}^{S} \alpha_i(n) \cdot a_{ij} b_{jY_{n+1}} \cdot \beta_j(n+1)}. \tag{13}$$

Finally, given an observation sequence Y and summing $\gamma_i(n)$ and $\xi_{ij}(n)$ over time (i.e. over the entire sequence length T) yields the following results:

1. Given Y, $\sum_{n=1}^{T} \gamma_i(n)$ is the expected number of visits to state i and, conversely, is also the expected number of transitions away from state i.
2. Given Y, $\sum_{n=1}^{T-1} \xi_{ij}(n)$ is the expected number of transitions from state i to state j.
3. The expected relative frequency spent in state i at time $n = 1$ forms an update to the initial hidden state distribution, π,

$$\hat{\pi}_i = \gamma_i(1). \tag{14}$$

4. The expected number of transitions from state i to state j relative to the expected number of transitions away from state i forms an update to the hidden state transition matrix A,

$$\hat{a}_{ij} = \frac{\sum_{n=1}^{T-1} \xi_{ij}(n)}{\sum_{n=1}^{T} \gamma_i(n)}. \tag{15}$$

5. The expected number of times the observation i is observed while in state j relative to the expected number of visits to state j forms an update to the observation distribution matrix B,

$$\hat{b}_{ji} = \frac{\sum_{n=1}^{T} \delta_{Y_n=i} \gamma_i(n)}{\sum_{n=1}^{T} \gamma_i(n)}. \tag{16}$$

For an initial parameterization of the model λ_0 and for a given an observation sequence Y, updating the model using the above equations yields a new model $\hat{\lambda}$ that is more likely than λ_0 to have produced the observation sequence. By iteratively applying the update equations, Baum *et al.* have shown that the model achieves a local maximum in the likelihood function of the parameterized model given the observation information [23, 24].

2.1.3.7 Example Problem

Given a two-state, discrete HMM, $\lambda = (A, B, \pi)$, parameterized by the hidden state transition matrix

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, \quad \text{where} \quad [a_{ij}] = P\{X_{n+1} = j | X_n = i\}, \tag{17}$$

the observation distribution matrix

$$B = \begin{bmatrix} 0.89 & 0.11 \\ 0.11 & 0.89 \end{bmatrix}, \quad \text{where} \quad [b_{ji}] = P\{Y_n = i | X_n = j\}, \tag{18}$$

and the initial state distribution vector

$$\pi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad \text{where} \quad [\pi_i] = P\{X_0 = i\}, \tag{19}$$

find $\gamma_i(n) = P\{X_n = i | Y, \lambda\}$, defined as the probability of being in state i at time n given the observation sequence

$$Y = (2, 2, 2, 1, 1, 2, 2, 2)$$

and the above model $\lambda = (A, B, \pi)$. First, note that $\gamma_i(n)$ can be written in terms of the forward and backward variables

$$\gamma_i(n) = \frac{\alpha_i(n) \cdot \beta_i(n)}{\sum_{j=1}^{S} \alpha_j(n) \cdot \beta_j(n)}.$$

Second, determine the best estimate of the sequence of hidden states given the observation sequence Y. The forward variable is defined as $\alpha_j(n+1) = \left[\sum_{i=1}^{S} \alpha_i(n) a_{ij}\right] b_{jY_{n+1}}$. Table 1 iterates through the forward variable calculations.

The backward variable is defined as $\beta_i(n) = \sum_{j=1}^{S} a_{ij} b_{jY_{n+1}} \beta_j(n+1)$. Table 2 iterates through the backward variable calculations.

To find the probability of being in state i at time n given the model λ and the observation sequence $Y = (2, 2, 2, 1, 1, 2, 2, 2)$, the forward and backward variable

Table 1. Forward variable calculations

n	$\alpha_1(n)$	$\alpha_2(n)$
0	$= 0.5$	$= 0.5$
1	$\big[0.9 \cdot \alpha_1(0) + 0.1 \cdot \alpha_2(0)\big] \cdot 0.11 = 0.055$	$\big[0.1 \cdot \alpha_1(0) + 0.9 \cdot \alpha_2(0)\big] \cdot 0.89 = 0.455$
2	$\big[0.9 \cdot \alpha_1(1) + 0.1 \cdot \alpha_2(1)\big] \cdot 0.11 = 0.0103$	$\big[0.1 \cdot \alpha_1(1) + 0.9 \cdot \alpha_2(1)\big] \cdot 0.89 = 0.3613$
3	$\big[0.9 \cdot \alpha_1(2) + 0.1 \cdot \alpha_2(2)\big] \cdot 0.11 = 0.005$	$\big[0.1 \cdot \alpha_1(2) + 0.9 \cdot \alpha_2(2)\big] \cdot 0.89 = 0.2904$
4	$\big[0.9 \cdot \alpha_1(3) + 0.1 \cdot \alpha_2(3)\big] \cdot 0.89 = 0.0299$	$\big[0.1 \cdot \alpha_1(3) + 0.9 \cdot \alpha_2(3)\big] \cdot 0.11 = 0.0288$
5	$\big[0.9 \cdot \alpha_1(4) + 0.1 \cdot \alpha_2(4)\big] \cdot 0.89 = 0.0265$	$\big[0.1 \cdot \alpha_1(4) + 0.9 \cdot \alpha_2(4)\big] \cdot 0.11 = 0.0032$
6	$\big[0.9 \cdot \alpha_1(5) + 0.1 \cdot \alpha_2(5)\big] \cdot 0.11 = 0.0027$	$\big[0.1 \cdot \alpha_1(5) + 0.9 \cdot \alpha_2(5)\big] \cdot 0.89 = 0.0049$
7	$\big[0.9 \cdot \alpha_1(6) + 0.1 \cdot \alpha_2(6)\big] \cdot 0.11 = 0.00032$	$\big[0.1 \cdot \alpha_1(6) + 0.9 \cdot \alpha_2(6)\big] \cdot 0.89 = 0.0042$
8	$\big[0.9 \cdot \alpha_1(7) + 0.1 \cdot \alpha_2(7)\big] \cdot 0.11 = 7.7 \cdot 10^{-5}$	$\big[0.1 \cdot \alpha_1(7) + 0.9 \cdot \alpha_2(7)\big] \cdot 0.89 = 0.0034$

Table 2. Backward variable calculations

n	$\beta_1(n)$	$\beta_2(n)$
8	$= 1$	$= 1$
7	$0.9 \cdot 0.11\beta_1(8) + 0.1 \cdot 0.89\beta_2(8) = 0.188$	$0.1 \cdot 0.11\beta_1(8) + 0.9 \cdot 0.89\beta_2(8) = 0.812$
6	$0.9 \cdot 0.11\beta_1(7) + 0.1 \cdot 0.89\beta_2(7) = 0.0909$	$0.1 \cdot 0.11\beta_1(7) + 0.9 \cdot 0.89\beta_2(7) = 0.6525$
5	$0.9 \cdot 0.11\beta_1(6) + 0.1 \cdot 0.89\beta_2(6) = 0.0671$	$0.1 \cdot 0.11\beta_1(6) + 0.9 \cdot 0.89\beta_2(6) = 0.5236$
4	$0.9 \cdot 0.89\beta_1(5) + 0.1 \cdot 0.11\beta_2(5) = 0.0595$	$0.1 \cdot 0.89\beta_1(5) + 0.9 \cdot 0.11\beta_2(5) = 0.0578$
3	$0.9 \cdot 0.89\beta_1(4) + 0.1 \cdot 0.11\beta_2(4) = 0.0483$	$0.1 \cdot 0.89\beta_1(4) + 0.9 \cdot 0.11\beta_2(4) = 0.0110$
2	$0.9 \cdot 0.11\beta_1(3) + 0.1 \cdot 0.89\beta_2(3) = 0.00576$	$0.1 \cdot 0.11\beta_1(3) + 0.9 \cdot 0.89\beta_2(3) = 0.00936$
1	$0.9 \cdot 0.11\beta_1(2) + 0.1 \cdot 0.89\beta_2(2) = 0.0014$	$0.1 \cdot 0.11\beta_1(2) + 0.9 \cdot 0.89\beta_2(2) = 0.0075$

calculations of Tables 1 and 2 are used to calculate

$$\gamma_i(n) = P\{X_n = i|Y, \lambda\} = \frac{\alpha_i(n) \cdot \beta_i(n)}{\sum_{j=1}^{S} \alpha_j(n) \cdot \beta_j(n)}.$$

Table 3 shows the resulting probabilities with boldface indicating the more likely state at time n. Thus, the best estimate of the sequence of hidden states given the observation sequence is

$$\{X_n\}_1^8 = (S_2, S_2, S_2, S_1, S_1, S_2, S_2, S_2).$$

Table 3. Gamma variable calculations

n	$\gamma_1(n)$	$\gamma_2(n)$
1	0.02243	**0.97757**
2	0.01731	**0.98269**
3	0.07015	**0.92985**
4	**0.51604**	0.48396
5	**0.51604**	0.48396
6	0.07015	**0.92985**
7	0.01731	**0.98269**
8	0.02243	**0.97757**

2.1.3.8 Learning Problem using Expectation Maximization (EM)

In this section an EM approach is taken toward parameter re-estimation of a discrete HMM. The goal is to maximize the likelihood (or in this case log-likelihood) function $\mathcal{L}$ by finding the maximum likelihood estimates (MLE) of the model parameters λ given the complete data (i.e., the observation data Y and the hidden state sequence X). The likelihood function with complete data is

$$\mathcal{L}(\lambda) = \log P(Y, X|\lambda).$$

However, we have incomplete data in the case of the hidden Markov model: we do not know the true hidden state sequence, X. We seek to maximize the posterior probability of the parameters λ given the observation data Y, marginalizing over the missing state sequence data X:

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}} \log \left(\sum_X P(Y, X|\lambda) \right).$$

Finding the MLE for λ by maximizing $\mathcal{L}(\lambda)$ directly can be difficult to compute (log of a large sum). A simplification makes use of Jensen's inequality, which states

$$E\left[f(X)\right] \leq f(E[X])$$

for X a random variable and f a concave (e.g. log) function defined over at least the range of X, which changes the log of large sum to a sum of logs. Thus, given an arbitrary distribution $F(X)$ over the hidden variables

$$\begin{aligned} \log \sum_X P(Y, X|\lambda) &= \log \sum_X F(X) \frac{P(Y, X|\lambda)}{F(X)} && (20) \\ &\geq \sum_X F(X) \log \frac{P(Y, X|\lambda)}{F(X)} && (21) \\ &= \sum_X F(X) \log P(Y, X|\lambda) - \sum_X F(X) \log F(X) && (22) \\ &= Q(\lambda, F). && (23) \end{aligned}$$

The EM algorithm [25] provides an iterative approximation method which alternates between maximizing Q with respect to F while holding λ fixed and maximizing Q with respect to λ while holding F fixed:

1. Set $p = 0$ and choose λ_p, the initial HMM parameter estimates.
2. Perform the expectation step:

$$F_{p+1} \leftarrow \operatorname*{argmax}_F Q(\lambda_p, F) \tag{24}$$

3. Perform the maximization step:

$$\lambda_{p+1} \leftarrow \operatorname*{argmax}_\lambda Q(\lambda, F_{p+1}) \tag{25}$$

4. Replace p with $p + 1$ and repeat steps 2 through 4 until a stopping threshold is reached.

   Finding F_{p+1} in step 2 begins with

$$Q(\lambda_p, F) = \sum_X F(X) \log P(Y, X|\lambda_p) - \sum_X F(X) \log F(X) \tag{26}$$

and maximizing over F. To do this the Lagrangian

$$\tilde{Q} = Q + \gamma\,(1 - \sum_X F(X)),$$

is introduced. Determining its derivative

$$\frac{\partial \tilde{Q}}{\partial F} = \log P(Y, X|\lambda_p) - \log F(X) - 1 + \gamma$$

and setting it equal to zero yields

$$\begin{aligned} 0 &= \log P(Y, X|\lambda_p) - \log F(X) - 1 + \gamma \\ \log F(X) &= \log P(Y, X|\lambda_p) - 1 + \gamma \\ F(X) &= P(Y, X|\lambda_p) \cdot e^{\gamma - 1}. \end{aligned} \tag{27}$$

Summing over X yields

$$\begin{aligned} \sum_X F(X) = 1 &= e^{\gamma-1} \sum_X P(Y, X|\lambda_p) \\ e^{\gamma-1} &= \frac{1}{P(Y|\lambda_p)}, \end{aligned} \tag{28}$$

and substituting Eq. 28 into Eq. 27 gives

$$F(X) = \frac{P(Y, X|\lambda_p)}{P(Y|\lambda_p)} = P(X|Y, \lambda_p). \tag{29}$$

When F_{p+1} is set to $P(X|Y,\lambda_p)$, Eq. 26 becomes

$$\begin{aligned}
Q(\lambda_p, F_{p+1}) &= Q(\lambda_p, P(X|Y,\lambda_p)) \\
&= \sum_X P(X|Y,\lambda_p)\log P(Y,X|\lambda_p) - \sum_X P(X|Y,\lambda_p)\log P(X|Y,\lambda_p) \\
&= \sum_X P(X|Y,\lambda_p)\cdot\log\frac{P(Y,X|\lambda_p)}{P(X|Y,\lambda_p)} \\
&= \sum_X P(X|Y,\lambda_p)\cdot\log P(Y|\lambda_p) \\
&= \log P(Y|\lambda_p)\cdot\sum_X P(X|Y,\lambda_p) \\
&= \log P(Y|\lambda_p)\cdot 1 \\
&= \log P(Y|\lambda_p) \\
&= \mathcal{L}(\lambda_p). \qquad (30)
\end{aligned}$$

Thus, the maximum in step 2 is obtained by setting $F_{p+1}(X) = P(X|Y,\lambda_p)$, where the bound becomes an equality with the objective $Q(\lambda_p, F_{p+1}) = \mathcal{L}(\lambda_p)$. The maximum in step 3 is found by maximizing the first term of Eq. 22, since the second term does not depend on λ. Thus

$$\lambda_{p+1} \leftarrow \underset{\lambda}{\operatorname{argmax}} \sum_X P(X|Y,\lambda_p)\log P(Y,X|\lambda) \qquad (31)$$

The sequence $\lambda_0, \lambda_1, \ldots, \lambda_p$, for $p \geq 0$, yields nondecreasing values of the likelihood function that converge to a local maximum. Thus Q forms a lower bound of the likelihood function $\mathcal{L}$. The EM algorithm ascends the likelihood function in the parameter space.

In the following steps we evaluate Eq. 31 by summing over all $X \in \{X_T\}$ to define the incomplete-data Q function in terms of the complete-data:

$$Q(\lambda, F_{p+1}) = \sum_X P(X|Y,\lambda_p)\log P(Y,X|\lambda) \qquad (32)$$

Given a particular state sequence $X = x_0 x_1 \ldots x_T$,

$$P(Y, X|\lambda) = \pi_{x_0} \prod_{n=1}^{T} a_{x_{n-1}x_n} b_{x_n y_n}.$$

Substituting back into the Q function, and simplifying yields

$$\begin{aligned}
Q(\lambda, F_{p+1}) &= \sum_X P(X|Y, \lambda_p) \log \left(\pi_{x_0} \prod_{n=1}^{T} a_{x_{n-1}x_n} b_{x_n y_n} \right) \\
&= \sum_X \log \pi_{x_0} \cdot P(X|Y, \lambda_p) + \sum_X \left[\sum_{n=1}^{T} \log a_{x_{n-1}x_n} b_{x_n y_n} \right] \cdot P(X|Y, \lambda_p) \\
&= \sum_X \log \pi_{x_0} \cdot P(X|Y, \lambda_p) + \sum_X \left[\sum_{n=1}^{T} \left(\log a_{x_{n-1}x_n} + \log b_{x_n y_n} \right) \right] \cdot P(X|Y, \lambda_p) \\
&= \sum_X \log \pi_{x_0} \cdot P(X|Y, \lambda_p) + \sum_X \left[\sum_{n=1}^{T} \log a_{x_{n-1}x_n} \right] \cdot P(X|Y, \lambda_p) \; + \; \cdots \\
&\quad \sum_X \left[\sum_{n=1}^{T} \log b_{x_n y_n} \right] \cdot P(X|Y, \lambda_p). \qquad (33)
\end{aligned}$$

Taking each term of Eq. 33 in turn, the parameters of the HMM are optimized. Taking the first term and finding the marginal expression at time $n = 0$ gives

$$\sum_X \log \pi_{x_0} \cdot P(X|Y, \lambda_p) = \sum_{i=1}^{S} \log \pi_i \cdot P(x_0 = i|Y, \lambda_p),$$

where S is the number of hidden states. To optimize, a Lagrange multiplier γ is used and the added stochastic constraint $\sum_{i=1}^{S} \pi_i = 1$ is enforced:

$$\frac{\partial}{\partial \pi_i} \left(\sum_{i=1}^{S} \log \pi_i \cdot P(x_0 = i|Y, \lambda_p) + \gamma \left(\sum_{i=1}^{S} \pi_i - 1 \right) \right) = 0.$$

Solving for π_i yields

$$\pi_i = P(x_0 = i|Y, \lambda_p). \qquad (34)$$

The second term of Eq. 33 becomes

$$\sum_X \left[\sum_{n=1}^{T} \log a_{x_{n-1}x_n} \right] \cdot P(X|Y, \lambda_p) = \sum_{i=1}^{S} \sum_{j=1}^{S} \sum_{n=1}^{T} \log a_{ij} P(x_{n-1} = i, x_n = j|Y, \lambda_p).$$

Applying a Lagrange multiplier and the constraint $\sum_{j=1}^{S} a_{ij} = 1$ and solving yields

$$a_{ij} = \frac{\sum_{n=1}^{T} P(x_{n-1} = i, x_n = j|Y, \lambda_p)}{\sum_{n=1}^{T} P(x_{n-1} = i|Y, \lambda_p)}. \tag{35}$$

The third term of Eq. 33 becomes

$$\sum_X \left[\sum_{n=1}^{T} \log b_{x_n y_n} \right] \cdot P(X|Y, \lambda_p) = \sum_{j=1}^{S} \sum_{n=1}^{T} \log b_{j y_n} P(x_n = j|Y, \lambda_p).$$

Applying a Lagrange multiplier and the constraint $\sum_{i=1}^{Q} b_{ji} = 1$ (where Q is the size of the discrete alphabet) and solving yields

$$b_{ji} = \frac{\sum_{n=1}^{T} P(x_n = j|Y, \lambda_p) \delta_{Y_n = i}}{\sum_{n=1}^{T} P(x_n = j|Y, \lambda_p)}, \tag{36}$$

where the δ-function contributes only when the n^{th} observation matches the i^{th} symbol of the observation alphabet. Note that the parameter re-estimation equations of the EM development (Eqns. 34, 35, and 36) match those of the previous development (Eqns. 14, 15, and 16) with the subtle difference of indexing time at $n = 0$ to T instead of $n = 1$ to $T + 1$.

2.1.3.9 Extension to Continuous Observation Space

To this point, development of HMM theory uses a discrete observation space, i.e., a discrete probability density associated with each hidden state models the observations. For problems where observations are continuous signals, a method must be used to quantize the signals into a discrete space. This process may degrade

model performance by losing information through quantizing. A useful extension of the discrete HMM is one with continuous observation densities.

Previously, the observation distribution matrix B is defined by $[b_{ji}] = P\{Y_n = i|X_n = j\}$, where each observation Y_n of the sequence $\{Y_n, n \geq 0\}$ is one of Q possible values. In the continuous case, researchers typically use a finite mixture of Gaussians to approximate any finite, continuous density function [18]. A continuous observation probability with M mixture components and S states has the form

$$b_j(Y) = \sum_{k=1}^{M} c_{jk}\Psi(\mu_{jk}, \Sigma_{jk}) \quad \text{for} \quad 1 \leq j \leq S, \tag{37}$$

where Y is the observation sequence, c_{jk} is the k^{th} component mixture in state j, and Ψ is the Gaussian kernel for the k^{th} component mixture in state j with mean μ_{jk} and covariance Σ_{jk}. The kernel is

$$\Psi(\mu_{jk}, \Sigma_{jk}) = \frac{1}{(2\pi)^{d/2}|\Sigma_{jk}|^{1/2}} \exp\left[-\frac{1}{2}(x - \mu_{jk})'\Sigma_{jk}^{-1}(x - \mu_{jk})\right]. \tag{38}$$

The following constraints must be satisfied by the mixture components:

$$\sum_{k=1}^{M} c_{jk} = 1 \quad \text{for} \quad 1 \leq j \leq S \tag{39}$$

$$c_{jk} \geq 0 \quad \text{for} \quad 1 \leq j \leq S, \quad 1 \leq k \leq M, \tag{40}$$

leading to a probability density function that integrates to one over the observations

$$\int_{-\infty}^{\infty} b_j(Y)dY = 1 \quad \text{for} \quad 1 \leq j \leq S. \tag{41}$$

2.1.3.10 HMMs in Automatic Target Recognition

This section reviews applications in the literature of HMMs to target recognition, specifically using high range-resolution radar (HRR) signatures as the source of

classifier feature data. The purpose is to highlight encouraging results while noting the assumptions and limitations of each experiment.

A series of Air Force Institute of Technology (AFIT) research efforts [42, 17, 43] applied HMMs to pattern recognition problems using features based on HRR signatures. DeWitt [42] processed HRR signatures produced by a synthetic CAD-based Xpatch® model using the Prony technique. The feature vectors produced by the Prony technique describe scattering centers of the target. These feature vectors were quantized using a k-means method in order to apply a discrete HMM. DeWitt considered a two-class problem with prior knowledge of target aspect and azimuth to within $\pm 5^{\circ}$. To test classifier robustness, Gaussian noise was added to the training data.

Fielding [17] compared discrete and Gaussian-mixture HMMs in an effort to classify sequences of 2-D images of 3-D objects. A five-class problem of ground targets with additive noise was studied. Feature data was derived from the coefficients of low-frequency Fourier transformed CAD-based target images. Prior knowledge of target aspect angle was $\pm 45^{\circ}$. In the discrete case a clustering method was used to quantize the data. Fielding found that the continuous HMM performed better than the discrete HMM in general but not at all experiment design points.

MacDonald [43] applied Gaussian-mixture HMMs operating on low-frequency spectral components of Fourier transformed HRR signatures. He studied a three-class problem of airborne targets. The research found that forcing a relationship between the hidden states and target orientation improved classification performance. The process resulted in an observable Markov process rather than a hidden one. It was unclear how training and testing data were segregated (if at all).

Another series of inter-related research, separate from the above listed AFIT research, focused on HMM-based time-series classification [44, 45, 46, 47, 48]. Runkle compared discrete versus Gaussian-mixture HMMs in classifying submerged objects

using features extracted from sequences of acoustic waveforms and demonstrated the marked benefit of using continuous HMMs [44, 45].

Bharadwaj and Runkle applied continuous HMMs with linear density distributions in the observed feature space [46]. The study used two airborne targets modeled by Xpatch® with features extracted via matching pursuits.

Liao and Runkle applied HMMs to ground target identification using features extracted from SAR-based HRR signatures [47, 48]. In both papers the RELAX algorithm [49] was used to extract point scatterer features from HRR signatures of sequenced SAR data of ten target classes from the Moving and Stationary Target Acquisition and Recognition (MSTAR) data collection (covered in Section 2.2).

Other research has been reported in the area of HMMs using HRR signatures for target classification. Paul implemented a hybrid classifier using an eigen-template to score HRR signatures prior to being input to discrete HMM classifiers [50]. His study used MSTAR data with four targets, but appears to have used the same data to train and test the hybrid classifier.

In Kottke et al. [51] and Nilubol et al. [52, 53] a Radon transformation on segmented two-dimensional SAR images was used to produce rotation and translation-independent features. These features were ordered, clustered, and input to class-specific discrete HMMs for classification.

Additionally, Jacobs et al. [54], Zhou et al. [55], and Pei et al. [56] each implemented HMM-based classifiers acting on sequenced HRR signatures.

Evidence of success in applying HMMs to the problem of sequential observation target classification warrants further study. A review of the literature shows several areas of potential research:

- use collected SAR data instead of synthetic data to best capture realistic operating conditions
- use multi-class target sets with greater than five target classes

- use a methodology to design HMM structures that is supported by model selection and information theory
- include a rejection option for classifier labeling
- study of out-of-library performance for HMMs
- study the impact of prior knowledge of initial target pose
- study classifier performance when constrained by warfighter preferences

2.2 High range-resolution radar

2.2.1 Introduction

Target recognition of moving targets based on SAR imaging poses a challenge due to blurring from target motion while forming the synthetic aperture. Recent research [57, 58, 59] points to high range-resolution radar (HRR) as a possible approach for recognizing moving targets. Here, HRR refers to a radar operating in a specified bandwidth that is capable of producing high-resolution returns with significantly enhanced target to clutter (and noise) ratios through Doppler filtering and clutter cancellation. Returns from HRRs form focused range (or one-dimensional) profiles which identify specific target scattering centers. These scattering centers are related to the physical geometry and material composition of the target and thus form a means of identifying the target.

2.2.2 Literature

Several AFIT research efforts have studied HRR signatures and their use in target classification. In addition to DeWitt[42] and MacDonald[43] as described in Section 2.1.3.10, Meyer's PhD research [60] studied invariant features drawn from sequenced HRR signatures and applied a template-based classifier.

Zumwalt's master's thesis [61] used XPatch®-derived HRR signatures of airborne targets as the feature source. Zumwalt proposed a multinomial pattern matching classifier which out-performed baseline linear and quadratic classifiers.

HRR-based target recognition research outside of AFIT includes Williams [57, 58, 59] and proposes template-based ATR algorithms using HRR-derived features. Mitchell [62] introduced a statistical feature based classifier acting on HRR profiles of airborne targets. Shaw [63] used a template-based classifier with eigenvalues associated with HRR profiles across aspect angle. Zajic [64] employed wavelets-based features drawn from HRR profiles in a template scheme.

The research performed and reported here combines HMMs with HRR signatures in a classification experiment. Fundamentally, an airborne radar illuminates a ground target and the reflected radar information is collected and processed for classification. Previous efforts have used features derived from HRR profiles in classifying airborne and ground-based targets, but fusing multiple HMMs operating on HRR-derived features breaks new ground.

2.2.3 HRR Processing

This research uses complex SAR data contained in two collections, MSTAR [65] and DCS. Processing of the SAR data is required to form HRR signatures. This section describes the required steps.

- The target in the SAR chip is segmented (outlined) from background clutter using a target-sized mask to simulate doppler filtering (MSTAR contains stationary targets).
- The SAR image formation process in the cross-range dimension is reversed.
- Cross-range inverse FFTs are applied to obtain range signatures collected over the synthetic aperture.

- The complex range/angle data is de-weighted in angle using an inverse Taylor window over the valid data.
- Each range bin is magnitude-detected and normalized by the mean power in the signature to remove automatic gain control and range effects.
- The pixels are averaged in azimuth to form the HRR profile.

Of the SAR imagery used in the research of Section 2.1.3.10, the most realistic studies used HRR signatures derived from MSTAR SAR data.

2.2.4 MSTAR Program

The conversion process begins with the SAR chip. The example chip is taken from the MSTAR publicly-available data set, a subset of Collection 1 taken Sep 1995 at the Redstone Arsenal, Huntsville, AL by the Sandia National Laboratory (SNL) STARLOS sensor, operating at X-band in one foot resolution spotlight mode. The collection was jointly sponsored by DARPA and AFRL as part of the MSTAR program.

The example SAR chip is of a T-72 main battle tank, serial number 812 (1 of 3 T-72 tanks imaged in the publicly-available data set), 17 degree angle of depression, and an aspect angle of 345.8 degrees. Figure 7 shows a photograph of the T-72 target.

2.2.5 SAR Chip

This section discusses MSTAR target chip image files. Target chips are sub-images extracted from MSTAR target-type full scene images. MSTAR target chips consist of an ASCII Phoenix header followed by a section of 32-bit floating point magnitude data and a section of 32-bit floating point phase data (in polar complex format). Target chip image data is calibrated in units of meters for magnitude data

Figure 7. Photograph of the target T-72 main battle tank.

and radians for phase data. Figure 8 shows the raw magnitude and phase chip data in grayscale form.

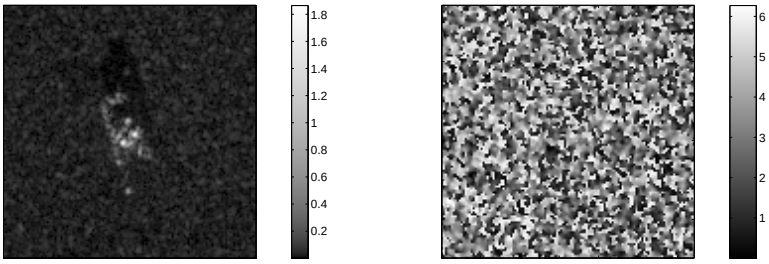

Figure 8. Magnitude (on the left) and phase (on the right) information from the example T-72 MSTAR SAR chip. Both are 128 by 128 pixels in size. Pixel information is shown through a 256-level grayscale.

The Phoenix header is the standard ASCII data header included with all MSTAR image files. MSTAR target chip Phoenix headers contain general and sensor-specific information. Table 4 contains the complete header information from the example MSTAR SAR chip.

Table 4. MSTAR SAR chip header information

PhoenixHeaderLength:	1975
PhoenixSigSize:	133047
PhoenixSigNum:	1
PhoenixHeaderCallingSequence:	''
HeaderVersionNumber:	'2CM'
native_header_length:	0
Filename:	'hb03787.0016'
Chip_MD5_CheckSum:	'd721d2b842fe498 a9f3ccb67c797fac9'
ParentScene:	'hb03787'
Site:	'redstn'
NumberOfColumns:	128
NumberOfRows:	128
TargetType:	't72_tank'
TargetSerNum:	'812'
TargetAz:	345.7742
TargetRoll:	-0.5911
TargetPitch:	359.6368
TargetYaw:	35.0758
DesiredDepression:	17
DesiredGroundPlaneSquint:	-90
DesiredSlantPlaneSquint:	-90
DesiredRange:	4500
DesiredAimpointLat:	34.6781
DesiredAimpointLong:	86.6874
DesiredAimpointElevation:	166
DesiredAimpointLatRef:	'N'
DesiredAimpointLongRef:	'W'
MeasDepression:	17.0938
MeasGroundPlaneSquint:	-91.5775
MeasSlantPlaneSquint:	-91.5078
MeasuredRange:	4475
MeasAimpointLat:	34.6781
MeasAimpointLong:	273.3092
MeasAimpointElevation:	165.3860
MeasAimpointLatRef:	'N'
MeasAimpointLongRef:	'E'
MeasAntennaLat:	34.6533
MeasAntennaLong:	-86.6551
MeasAircraftHeading:	41.6016
MeasAircraftAltitude:	1.4808e+003
RadarMode:	'mode 5 - spot light'
SensorCalibrationFactor:	42.9960
RadarPosition:	'bottom'
Range3dBWidth:	0.3013
CrossRange3dBWidth:	0.3229
SceneCenterRefLine:	40
X_Velocity:	39.4570
DataCollectors:	'Sandia National Lab'
CollectionDate:	19950902
CollectionTime:	82205
CollectionName:	'hb'
SensorName:	'Twin Otter'
Classification:	'UNCLASSIFIED'
MultiplicativeNoise:	'-10 dB'
AdditiveNoise:	'-32 to -34 dB'
CenterFrequency:	'9.60 GHz'
CrossRangeWeighting:	'-35dB_Taylor'
RangeWeighting:	'-35dB_Taylor'
DynamicRange:	'64 dB'
Bandwidth:	'0.591 GHz'
RangeResolution:	0.3047
CrossRangeResolution:	0.3047
RangePixelSpacing:	0.2021
CrossRangePixelSpacing:	0.2031
AverageImageCalFactor:	0.9708
Polarization:	'HH'
TargetSeasonalCover:	'only growing vegitation'
TargetWaterContent:	'dry'

2.2.6 SAR Chip Manipulation

The original complex SAR chip is formed by combining the 128 by 128 magnitude information with the 128 by 128 phase information:

$$C_{orig} = Me^{i \cdot P},$$

where M is the matrix containing magnitude information and P is the matrix containing phase information. Figure 9 shows the 128 by 128 pixel complex modulus $\sqrt{X^2_{real} + X^2_{imaginary}}$.

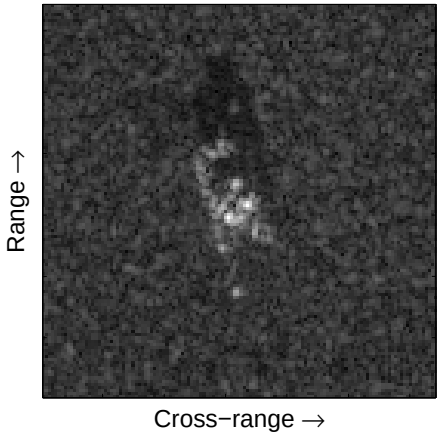

Figure 9. Combining the magnitude and phase information results in the baseline complex SAR chip.

To obtain the phase histories, or range profiles, several steps made in forming the MSTAR images are undone. The MSTAR images are formed by taking a 2-D inverse FFT of the Taylor-windowed, zero-padded phase history data on a rectangular grid. To undo these steps, the 2-D FFT of the 128 by 128 complex pixel chip described above is taken, then the transformed signal is shifted so that the small frequencies occur in the center. Figure 10 shows the resulting 2-D signal of the example MSTAR SAR chip.

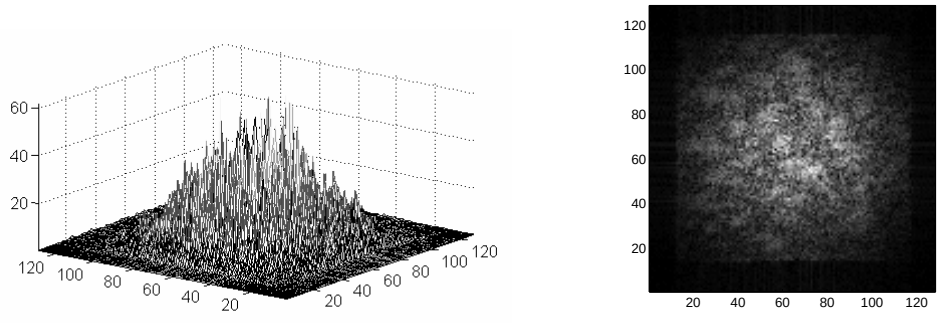

Figure 10. Mesh plot of the magnitude of the 2-D signal for the MSTAR sample chip. A grayscale image of the same signal is shown on the right.

A noticeable band of near-zero values appears at the border of the 2-D signal seen in Fig. 10. This band is assumed to be a result of zero-padding, and a 14 pixel wide band is removed from the perimeter of the signal, leaving the 100 by 100 signal shown in Fig. 11. Next, a process is undertaken to remove the Taylor windowing implemented when the SAR data is collected. MSTAR uses a 35 dB Taylor window with $\bar{n} = 4$. Figure 11 shows the described 2-D Taylor window.

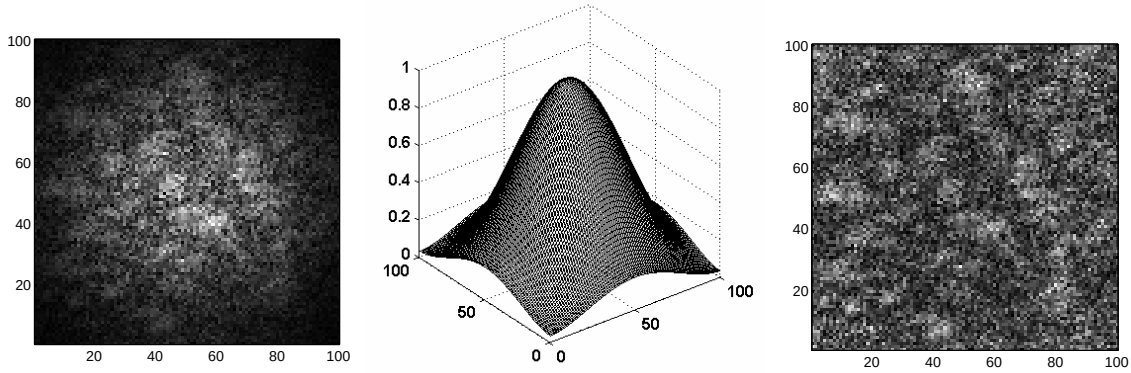

Figure 11. A cropped 100 by 100 grayscale image of the magnitude of the signal shown in Fig 10. In the middle is a 3-D plot of a Taylor window with 100 coefficients, a 35 dB sidelobe suppression level, and $\bar{n} = 4$, and on the right is the cropped signal with the windowing removed.

Finally, the cropped unwindowed signal information, also called the phase history, is processed by a 1-D FFT along the range dimension to reveal the range profiles

shown in Fig. 12. The magnitude of these range profiles and the mean profile are also plotted. Features are extracted from the 1-D mean range profile.

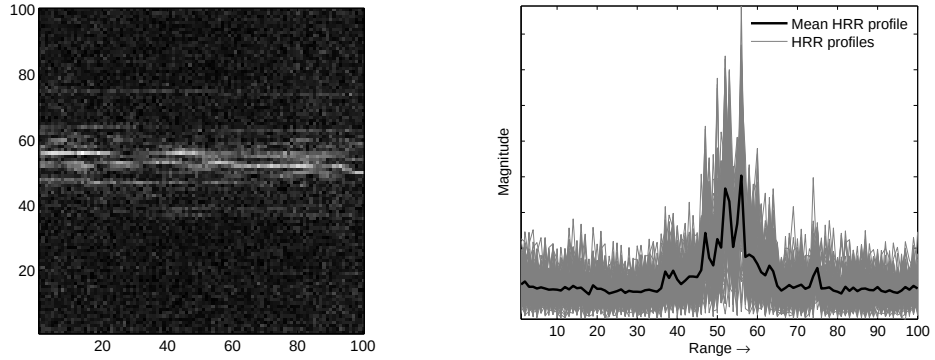

Figure 12. Taking a 1-D FFT of the cropped unwindowed signal of Fig. 11 results in the range profile information shown in the left plot. On the right, the individual range profiles (columns of left plot) are plotted in gray with the mean profile shown in black. Features are extracted from the mean profile.

2.3 Model selection

2.3.1 Introduction

One goal of the research is to develop a classifier using time-series models that explain the change in HRR signatures of moving targets. Selecting a model from among candidate models is the subject of this section. The choice should not be based solely on goodness-of-fit, but should also consider model complexity. An unnecessarily complex model may overfit a given set of data and generalize poorly. Model selection methods trade-off goodness-of-fit with model complexity in the search for the "*best*" model.

2.3.2 Literature

A review of the model selection literature yielded several survey papers [66, 67, 68] that treat the subject from a natural science perspective. The following

subsections on the various model selection techniques are derived from these sources and Burnham and Anderson's text [69].

Several papers specifically address estimation of the order of hidden Markov models. The order of an HMM refers to the number of states in the hidden Markov chain and is a measure of model complexity. Li et al. [70] use the Bayesian Information Criterion (BIC) to specify the order of an HMM in a handwriting-recognition application. Ryden [71] proposes a penalized likelihood estimator which can be used with the BIC and the Akaike Information Criterion (AIC) to estimate the order of an HMM. Ryden shows in a specified limit the estimator does not underestimate the order.

2.3.3 Likelihood criterion

Three aspects determine inference from models according to Fisher [72]: (1) model specification, (2) estimation of model parameters, and (3) estimation of precision. The Fisher likelihood theory assumes that the model specification is correct, leaving only the parameters of the model to be estimated. In cases such as linear regression models, the parameters can be estimated using maximum likelihood methods.

Suppose that a probability model g describes the probability distribution of the data x given the model parameters θ and a model specification or type g, i.e.,

$$g(x|\theta, model).$$

Also suppose that data is collected and a model is specified, but the model parameters are unknown. Then the likelihood function can be used for parameter estimation:

$$\mathcal{L}(\theta|x, model).$$

The probability model and likelihood model differ by what is known and what is sought. In the probability model, the parameters and the model are known. The probability of a given event (the data) is sought. In the likelihood model the data and the model are given, and estimates of the model parameters are sought. Thus the roles of the data and the parameters are reversed for the probability and likelihood models.

Burnham and Anderson [69] use a coin-flipping example to illustrate the likelihood concept. The experiment flips a coin n times and observes y heads. The coin is assumed to be unbiased and the coin flips are assumed to be independent. The binomial model is chosen to study the experiment. The likelihood function is

$$\mathcal{L}(p|y, n, binomial) = \binom{n}{y} p^y (1-p)^{n-y},$$

where p, the probability of a head, is the parameter of interest. One might calculate the likelihood of many values of p and pick the most likely one as the best estimate of p given the model. This is the *maximum likelihood estimator* and is found by maximizing

$$\log(\mathcal{L}(p|y, n, binomial)) = \log \binom{n}{y} + y \cdot \log(p) + (n-y) \cdot \log(1-p)$$

given n flips and y heads.

Likelihood ratio tests and maximum likelihood estimation are popular methods of parameter estimation. Standard statistics texts such as Wackerly [73], Mood [74], Hogg [75], and Bickel [76] treat the subject thoroughly.

2.3.4 Akaike's information criterion

Kullback and Leibler [77] introduced a "*distance*" metric to compare two models f and g,

$$I(f,g) = \int f(x) \log\left(\frac{f(x)}{g(x|\theta)}\right) \mathrm{d}x,$$

where $I(f,g)$ denotes the information lost when g is used to approximate f. The K-L distance is a fundamental quantity in information theory and is the basis for model selection paired with likelihood inference [69].

The K-L distance can be rewritten as

$$I(f,g) = \int f(x) \log(f(x)) \mathrm{d}x - \int f(x) \log(g(x|\theta)) \mathrm{d}x,$$

and, recognizing the two terms above as statistical expectations with respect to f, is

$$I(f,g) = \mathbf{E}_x[\log(f(x))] - \mathbf{E}_x[\log(g(x|\theta))].$$

Key to the relative comparison of two models is the assumption that f refers to the unknown "*true*" distribution and g is the approximating model. The "*true*" distribution f, while unknown, remains constant, and $\mathbf{E}_x[\log(f(x))]$ can be considered a constant C when calculating a relative distance between f and g:

$$I(f,g) = C - \mathbf{E}_x[\log(g(x|\theta))], \qquad \text{or} \qquad I(f,g) - C = -\mathbf{E}_x[\log(g(x|\theta))].$$

Now $(I(f,g) - C)$ becomes the relative distance between models f and g, making $(\mathbf{E}_x[\log(g(x|\theta))])$ a measure of interest for determining the best model. For instance,

given two models g_1 and g_2, if $I(f, g_1) < I(f, g_2)$, then g_1 is best. Also,

$$\begin{aligned} I(f, g_1) - C &< I(f, g_2) - C \\ \mathbf{E}_x[\log(g_1(x|\theta))] &< \mathbf{E}_x[\log(g_2(x|\theta))] \quad \text{and} \\ I(f, g_2) - I(f, g_1) &\equiv -\mathbf{E}_x[\log(g_2(x|\theta))] + \mathbf{E}_x[\log(g_1(x|\theta))]. \end{aligned}$$

Akaike's seminal work [78] introduced a method of model selection using K-L distance without the restriction of full knowledge of the "*true*" model f and the parameters θ. In Akaike's development the unique value of θ that minimizes the K-L distance $I(f, g)$ is unknown. At this value θ_0 information loss is minimized, and θ_0 is found as with the maximum likelihood estimator $\hat{\theta}$. Thus the model selection process shifts from minimizing known (θ_0) K-L distance to minimizing estimated K-L distance ($\hat{\theta}$) based on the expected value of the estimated parameters, or

$$\mathbf{E}_y \mathbf{E}_x \left[\log(g(x|\hat{\theta}(y))) \right],$$

where x and y are independent random samples from the same distribution and the expectations are taken with respect to truth (f). Akaike showed that using $\log(\mathcal{L}(\hat{\theta}|data))$, the maximized log-likelihood for each model g_i given data, to estimate K-L distance results in an upwardly biased estimate. He also showed that the bias can be corrected by incorporating the number of estimable parameters K, which can be considered a measure of complexity and hence a part of the classic tradeoff between bias and variance as a result of underfitting or overfitting data. However, Akaike's development finds K as a simple expression of the asymptotic bias in the log-likelihood as an estimator of $\mathbf{E}_y \mathbf{E}_x \left[\log(g(x|\hat{\theta}(y))) \right]$. Thus, $\log(\mathcal{L}(\hat{\theta}|data)) - K$ is an unbiased estimator of $\mathbf{E}_y \mathbf{E}_x \left[\log(g(x|\hat{\theta}(y))) \right]$.

For K-L distance in the general case,

$$I(f, g) - C = -\mathbf{E}_x[\log(g(x|\theta))],$$

and since the K-L distance is to be estimated using the MLE $\hat{\theta}$, the expectation of both sides yields

$$\mathbf{E}_y[I(f, g(x|\hat{\theta}(y))] - C = -\mathbf{E}_y\mathbf{E}_x[\log(g(x|\hat{\theta}(y)))].$$

Substituting the above result for the corrected estimator gives

$$\mathbf{E}_y[I(f, g(x|\hat{\theta}(y))] - C = -\log(\mathcal{L}(\hat{\theta}|data)) + K,$$

which, with rearrangement and inclusion of a factor of 2, yields the Akaike information criterion

$$\mathbf{AIC} = -2 \cdot \log(\mathcal{L}(\hat{\theta}|data)) + 2K. \tag{42}$$

2.3.5 Bayesian information criterion

Schwarz [79] presents an alternative to AIC,

$$\mathbf{BIC} = -2 \cdot \log \mathcal{L}(\hat{\theta}|data) + \log(n) \cdot K, \tag{43}$$

where n is the number of models being considered. The difference between AIC and BIC is the $\log(n)$ term.

The following derivation [69] shows the origin of the $\log(n)$ term. Given a model g_i the likelihood of parameter set θ_i (for K parameters θ is a vector of length K) given the data is $\mathcal{L}(\theta_i|x, g_i)$. The prior probability for θ_i is denoted $\pi_i(\theta_i)$. The marginal likelihood is

$$g_i(x) = \int g_i(x|\theta_i)\pi_i(\theta_i)\mathrm{d}\theta_i,$$

or the likelihood of model g_i given the data and the prior probability distribution of θ_i.

The marginal probability of the data is

$$\int \left[\prod_{j=1}^{n} g(x_i|\theta) \right] \pi(\theta) \mathrm{d}\theta,$$

which is

$$\int [\mathcal{L}(\theta|x, g)] \pi(\theta) \mathrm{d}\theta, \tag{44}$$

where x represents the data. As sample size increases the likelihood function near the maximum likelihood estimator $\hat{\theta}$ can be approximated as

$$\mathcal{L}(\theta|x, g) = \mathcal{L}(\hat{\theta}|x, g) \cdot \mathrm{e}^{-\frac{1}{2}(\theta-\hat{\theta})' V(\hat{\theta})^{-1} (\theta-\hat{\theta})},$$

where $V(\hat{\theta})$ is the estimated $K \times K$ variance-covariance matrix of the MLE. This form of the likelihood stems from the sampling distribution of the MLE becoming multivariate normal as the sample size goes to infinity. Substituting the estimated form of the likelihood back into Eq. 44 yields

$$\mathcal{L}(\hat{\theta}|x, g) \int \mathrm{e}^{-\frac{1}{2}(\theta-\hat{\theta})' V(\hat{\theta})^{-1} (\theta-\hat{\theta})} \pi(\theta) \mathrm{d}\theta.$$

As the sample size n goes to infinity, the approximation becomes exact, the likelihood concentrates near $\hat{\theta}$, and the prior is effectively uniform, so $\pi(\theta)$ can be treated as a constant. The integral is directly related to the underlying multivariate normal distribution

$$\int (2\pi)^{-K/2} \| V(\hat{\theta})^{-1} \|^{1/2} \mathrm{e}^{-\frac{1}{2}(\theta-\hat{\theta})' V(\hat{\theta})^{-1} (\theta-\hat{\theta})} \mathrm{d}\theta = 1,$$

where $\| \cdot \|$ is the determinant of a matrix. Using the normalizing constant yields

$$\int \left[\prod_{j=1}^{n} g(x_i|\theta) \right] \pi(\theta) \mathrm{d}\theta \approx \mathcal{L}(\hat{\theta}|x, g) \left[(2\pi)^{K/2} \| V(\hat{\theta})^{-1} \|^{-1/2} \right].$$

For a random sample, $V(\hat{\theta})^{-1} = nV_1(\hat{\theta})^{-1}$, where $V_1(\cdot)$ is independent of sample size. Also, $\|nV_1(\hat{\theta})^{-1}\| \equiv n^K \|V_1(\hat{\theta})^{-1}\|$. Thus

$$\int \left[\prod_{j=1}^{n} g(x_i|\theta)\right] \pi(\theta) \mathrm{d}\theta \approx \mathcal{L}(\hat{\theta}|x,g) \left[(2\pi)^{K/2} n^{-K/2} \|V(\hat{\theta})^{-1}\|^{-1/2}\right].$$

Taking -2 times the log of the right hand side yields the BIC criterion

$$-2\log(\mathcal{L}(\hat{\theta}|x,g)) + K\log(n) - K\log(2\pi) - \log(\|V_1(\hat{\theta})^{-1}\|).$$

The last two terms are dropped because they are dominated asymptotically by the order $\log(n)$ term and the order n log-likelihood term.

2.3.6 Method of cross-validation

The objective of cross-validation techniques is to evaluate model predictive accuracy [67]. The standard arrangement divides available data into a training set and a testing set. The training data is used to fit a model, resulting in a set of model parameters, $\hat{\theta}_{cal}$. Test data is used to measure the performance of the calibrated model.

2.4 Classifier fusion

2.4.1 Introduction

One goal of this research is the application of an architecture-selection methodology for the design of a multiple classifier system. This section outlines basic concepts and taxonomy associated with multiple classifier systems.

2.4.2 Literature

Multiple classifier system (MCS) literature can be divided into two groups: MCS theory and MCS application. Roli leads a continuing research effort in MCS theory; his fusion tutorial [10] is an excellent source for MCS concepts and taxonomy. Roli and Giacinto's book chapter [80] on design considerations for MCSs covers tradeoffs in design of the classifier ensemble and the fusing mechanism.

Combining outputs from a set of different classifiers is one method for the development of high performance classification systems. Roli and Giacinto believe that

> the rationale behind the growing interest in MCSs is that the classical approach to designing a pattern recognition system, which focuses on the search for the best individual classifier, has some serious drawbacks. The main drawback is that the best individual classifier for the classification task at hand is very difficult to identify, unless deep prior knowledge is available for such a task. [80]

One key concept in MCSs is that of complementary discriminatory power of classifiers. That is, the discriminatory information of one classifier may complement another classifier. Both classifiers make mistakes, but the mistakes are not identical, and so the combination of classifiers according to some rule will improve performance over the individual classifiers.

The design of an MCS can be split into two parts: first, design of the classifier ensemble, and second, design of the fusion function. The goal of the first part is to create a set of complementary, or diverse, classifiers. The goal of the second part is to create a mechanism that can exploit the complementary-ness of the classifiers and optimally combine them. The Roli fusion tutorial highlights several of these techniques [10].

Methods used to design the classifier ensemble assume a fixed decision function and generate a set of complementary classifiers to achieve the best accuracy relative

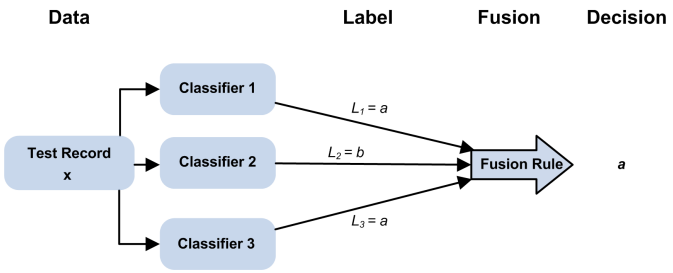

Figure 13. At the abstract level of fusion each classifier outputs a class label for each test record. A typical fusor is the majority vote scheme. Here three classifiers assign class membership to a test record and the decision rule chooses the final class membership.

to the decision function. Roli calls these methods coverage optimization methods, and some examples are [10]:

- injecting randomness into the classifier training algorithm, e.g. neural networks with different initializations
- manipulating training data by partitioning the data set or creating overlapping data sets
- manipulating input features, using feature selection methods and feeding different features to different classifiers
- manipulating output features, partitioning the set of classes in different ways, then assign classifiers to work on a subset of the whole class structure

Design of the combination function typically assumes a given set classifiers and has a goal of finding an optimal combination of decisions from those classifiers. Roli breaks down decision optimization into three groups: the abstract-level (see Fig. 13), the rank-level (see Fig. 14), and the measurement-level (see Fig. 15).

The Dasarathy short course on multi-sensor fusion [81] lists two fusion taxonomies: one based on sensor ensemble configuration or architecture, and one based on modes of input and output of the sensor ensemble. The first refers to how the multiple classifiers are connected, whether series, or parallel, or some combination of the two. The second taxonomy covers much the same ground as the Roli abstract–rank–measurement levels of fusion.

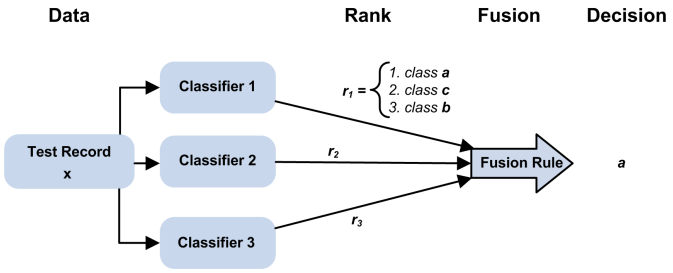

Figure 14. At the rank level of fusion each classifier outputs an ordered list of possible classes for each test record.

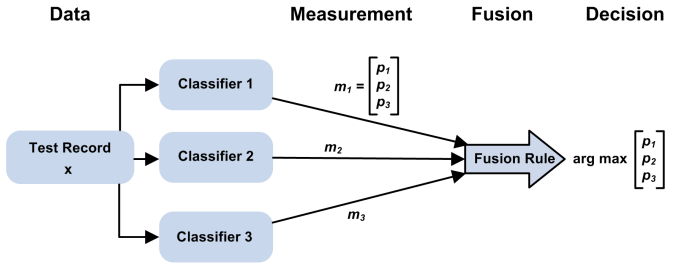

Figure 15. At the measurement level of fusion each classifier passes an output vector to the fusor. The fusor combines the multiple outputs across each vector element.

For Dasarathy the modes of fusion are divided into data-level, feature-level, and decision-level. Further, Dasarathy shows that sub-classes within this taxonomy are formed based on input to and output from the fuser. *Data In – Data Out* (DAI-DAO) fusion occurs when data from similar sensors are combined using arithmetic or logical operations; for instance, pixel intensities in multi-spectral image data. An example of *Features In – Features Out* (FEI-FEO) fusion is the fusion of two inputs, one an infrared sensor measuring cross-section, and the other a range radar measuring target depth. The fused output is a volumetric feature of the target. The most common fusion category is *Features In – Decision Out* (FEI-DEO). Here the recognition tool accepts features, then makes a classification decision. At the top level of fusion categories is *Decisions In – Decision Out* (DEI-DEO) fusion. Voting schemes fit into this category.

Several recent applications of MCS in the area of pattern recognition include Chan's fusion of dualband forward-looking infrared (FLIR) target data [82]. Other fusion applications include: Rizvi [83], which reports on various fusion techniques in a FLIR ATR application, and Song [84], which studies biomedical image identification using fused contextual information.

A series of AFIT master's research investigated fusion methods, correlation effects, and performance metrics [85, 86, 87, 88]. Storm [85] introduced a synthetic fusion testing environment and studied the effects of data correlation on three fusion techniques. Leap [86] extended Storm's work by examining the effects of sample size as well as correlation. Clemans [87] increased the number of classifiers in the ensemble to three from two and searched for the optimal ensemble given various experiment settings. Mindrup [88] extended Leap's work by allowing a non-declaration option from his classifiers, applying a cost function and finding the optimal fusion method.

A rejection, or non-declaration, parameter defines a region of class ambiguity where a classifier labels test records "unknown" [89]. A Bayes-optimal decision rule which assigns test records to the class with the maximum *a posteriori* probability

may be used. Rejection improves classification accuracy while decreasing misclassification errors by allowing the classifier to label "unknown" difficult-to-identify test records [89].

Using a rejection option creates a tradeoff between improved classification performance and the cost of gathering more information if a non-declaration is made. Chow [89] finds that the optimal rejection threshold given costs for misclassification, rejection, and correct classification are equivalent across data classes. Fumera et al. [11] apply class-specific rejection thresholds to account for varying class prior probabilities. Fumera proves that using multiple thresholds achieves equal or better classification performance than using the single rejection threshold of Chow.

Several authors use a loss function to set classification and rejection rules in a Bayes-optimal classification strategy (Chow [89], Devijver and Kittler [90], Fumera et al. [11] and Haspert [91]). By minimizing a loss function, classifier performance is optimized given set costs of rejection, classification errors, and correct classification in equivalent units. Setting the relative costs for classification error and rejection places the warfighter in the position of formally setting the cost of a fratricide incident versus non-declaration versus correct identification; a position the warfighter may not desire [4].

Laine's AFIT PhD research [15] presents a CID framework with a reject option that optimizes classification performance without resorting to a cost-based loss function.

2.5 Summary

This chapter presented relevant background for the investigation of an HMM-based MCS in a CID application, and HMM theory and application were described. Also, HRR signature processing and use in classification were covered, model selec-

tion theory was reviewed with specific attention to HMMs, and the fusion of multiple classifiers and rejection theory were reviewed.

3. HMM Classifier Development

3.1 Introduction

This chapter considers the development of HMM classifiers operating on HRR feature data. An introductory section illustrates the application of a simple HMM-based classifier to sequences of genetic data. This example supports the theory of the previous chapter.

Additionally, an implementation of model selection based on the theory presented in Chapter 2 using HMMs and synthetic data is presented.

3.2 Introductory HMM Classifier

This section presents an example of a discrete hidden Markov model used as a data sequence classifier. The classification of four-state genetic codes of varying lengths from two genetic groups, human and mouse, is investigated using discrete HMMs employing different numbers of hidden states. The impact on classification accuracy across numbers of hidden states is explored using test and validation data sets [92].

In this application the complexity of HMMs is explored using the number of hidden states. The hidden state space and observation state space are assumed to be fully-connected. Hence, the Markov chain may transition from any state to any state with probability greater than zero, and each state may produce any symbol from the discrete observation alphabet with probability greater than zero.

The hidden state transition matrix A and the observation distribution matrix B are initialized randomly before re-estimation using the Baum-Welch algorithm given training sequences. The implementation of the algorithm in MATLAB® uses code from Murphy's HMM toolkit [93].

3.2.1 Methodology

The initial concept of the project was to employ discrete HMMs to classify genetic sequences into one of two classes. Through research on biological sequencing in the Durbin's text [94], and online at the University of California at Santa Cruz's (UCSC) Computational Biology website [95], a satisfactory set of data was found.

The classification data consists of 77 randomly selected pairs of aligned genetic sequences of DNA from chromosome 10 of the mouse and human species. This data suits the purpose of this effort for a number of reasons. First, the genetic data describes two mammals and comes from identical locations on chromosome 10 of the respective DNA. Therefore, differences in the sequence data are a function of the species and not of the genetic location (either within the chromosome or across chromosomes). Second, the task of aligning the sequences has already been accomplished; roughly, this means each sequence is of the same length. Any disparity in length between the human and mouse pair is made up with space holders (later removed) such that sub-sequences within the larger sequences are aligned at mutually shared locations. Third, the data is naturally presented as a classification data set with one record for a human sequence and another for its aligned mouse partner.

A series of transforms produces a set of useable inputs for the MATLAB® HMM functions. The original data as downloaded from the UCSC's website consists of 77 paired sequences of human and mouse DNA in a flat text file; example sequences are shown in Fig. 16. The desired input to the MATLAB®-based HMM functions is a set of vectors representing the human sequence records and a set of vectors representing the mouse sequence records. To reach this goal, the data is separated into human and mouse data files, converted from text to integers, and converted into sub-sequences based on the location of the space-holding characters. There are 687 sub-sequences for each class of data.

```
1 chr10_random 877 890 chrX 100492763 100492776 - 825
TTTGTTTTTTAGTA
TTTTTTTTTTAACA

2 chr10_random 891 1054 chrX 119435191 119435335 - 7467
TAAAGAGTAGTATTTTATTGAATAAGATTTGCTCACAGAAAAATAAGCTTAAATCTGCAATGAATGCCAGACTCTACAGCAGAAAGCAATTTTCTCACTTTTCCACA
TAAAGAATAGTGCTTTATTGAATAAGTCTTATTCACAGAAAAATAAGCTTTAATCTACAACAAATGACAGATTATAGAGCAGAAAGCAATTC----------CA

3 chr10_random 19625 19788 chrX 27726436 27726580 + 7589
GTAGTACATTTGTGAATGAAATTTTATGGCTTTTTTCACTTAGTAGGAACCATTGTGTGTGGAAAAGTGAGAAAATTGCTTTCTGCTGTAGAGTCTGGCATTCATTG
GTACTATATTTGTGAATATAATTTTATGACTCGTTT------ATGGAAACTCCTTTATG----------GAATTGCTTTCTGCTCTATAATCTGTCATTTGTTG

4 chr10_random 19789 19802 chrX 46668995 46669008 + 825
TACTAAAAAACAAA
TGTTAAAAAAAAAA

5 chr10_random 36424 37047 chr13 111196092 111196743 + 28574
TCAGTCTTTGTG--TATAAAGACATCCATAGCAGGCTTT-ATCCAGCCAGCTTCTTTGGGATTCTTTATATGGTTTCAGGTCTATAGCATATCCACTAAAATATTCC
TCGATCTCTGAGACTAAAAAGGCATCCACAGCAGGCTTTTATCCAGCCAGCTTCTTTGAGACTCTTCATGGGTTTTGAGGTCTACAAAGTATACACTAATATACCCA

6 chr10_random 37048 37584 chr13 111221047 111221602 + 25447
TGTATGAGGATGATGTCCAGGATGTTGGTCTGGTGTCCCTGAGACAGCACTAACAGGTCCGTGGCTGGGTCCAGGTCCTTCCTGGACGGATTGGCAAGGAGCTCACT
TGAACATGGATGATGTCTGGGAAGTTGGCCAGGTGCCCCTGATACAGTGCTAACAGGTCCATGACTGGGTCCAGGTCCTGCCTGGGCTGCTCAGCGAAGAGTTCGCC

7 chr10_random 38489 38544 chr13 6093438 6093493 - 3062
CCAAGAAGTAGACCACAGGCCGTCTTGAGGAGGACTTTATGTTCAAGTGCAGAAAG
CTGAGAAGCAGGCCACCGGGCGCTTCGAAGAGGACTTCATCTCCAAACGCAGGAAG
```

Figure 16. Example gene sequences from chromosome 10 of human and mouse DNA.

The goal of the effort is to use HMMs, trained with sequences of known origin (human/mouse), to classify "unknown" sequences into either the human or mouse species. To accomplish this goal two HMMs are trained. One model is trained using human sequences and one model is trained using mouse sequences. The classification of a particular sequence results from a comparison of model likelihoods. Given a test sequence, if the human model is more likely than the mouse model to have produced the sequence, then the sequence is classified as human, with the converse true for a mouse sequence.

While classifying unknown sequences is the goal, insight into the relationship between model complexity and model performance is also sought. A series of experiments is devised to explore this relationship. Two HMMs with n hidden states are trained using 400 randomly-selected class-specific data records. For the experiment performed here, $n = [2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 15 \quad 20 \quad 30]$. Each HMM is tested using 100 randomly selected records from each class of data. Class membership is determined by comparing the likelihoods produced by the two class-specific HMM classifiers when presented with a test record.

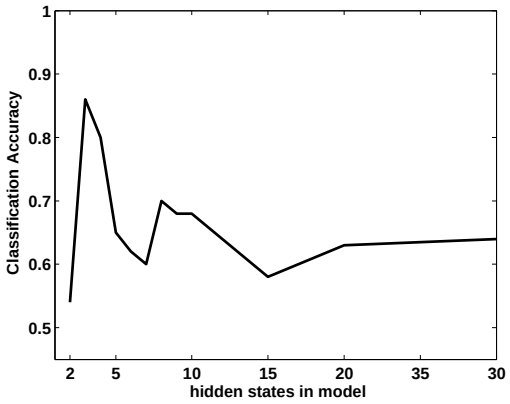

Figure 17. Plot of probability of correct selection versus number of hidden states.

3.2.2 Results

The experimental results (see Fig. 17) show that maximum classification accuracy is achieved with three hidden states in the models. Performance drops off rapidly as hidden states are added, indicating that a simple model (i.e., three hidden states) is preferable to a more complex model.

This experiment shows that HMMs form a useful tool for classifying sequenced discrete-valued data.

3.3 Model Selection with HMMs

This section investigates model complexity in HMMs using MATLAB®. First, discrete HMMs operating on discrete data are considered. Then, a comparison of discrete and continuous HMMs operating on continuous data is performed. Finally, a multi-variate Gaussian HMM is used to classify sequenced multi-dimensional data.

3.3.1 Complexity in Discrete HMMs

Discrete hidden Markov models have both a discrete state space and a discrete observation space. The following experiment examines measures of complexity in a

discrete HMM-based classifier. The experiment applies various measures of complexity to identify the most appropriate model given an ensemble of potential models. Data is generated from a stochastic model of known complexity. The data is then used to train and test a discrete HMM-based classifier. Based on classifier output, a model selection technique is applied. Then a comparison is made between the controlled user-defined data complexity and the suggested model complexity based on model outputs.

The experiment includes the following steps:

- Choose experiment parameters. Here the parameters of the stochastic model used to generate the data are specified.
- Generate training and testing data based on the parameter set
- Generate initial discrete HMMs of varying complexity
- Train the HMMs using a training data set
- Test the HMMs using a testing data set
- Reduce the HMM state space by one state and repeat the training/testing sequence

The output of an experiment is a mean log-likelihood achieved by averaging the log-likelihoods produced by the trained HMMs for each testing record. Thus as the number of testing records increases, the better the estimator (mean log-likelihood) for model performance.

Figure 18 shows the processes of the complexity experiment. The experiment begins with a highly-complex discrete HMM (20 hidden states) and the complexity is iteratively reduced by one state. The state to be removed is chosen based on its relative probability of in-transitioning. This choice of rule is arbitrary. The decision is made by summing over each column of the hidden state transition probability matrix. The column sums are compared and the state associated with the min value is chosen as the state to be removed.

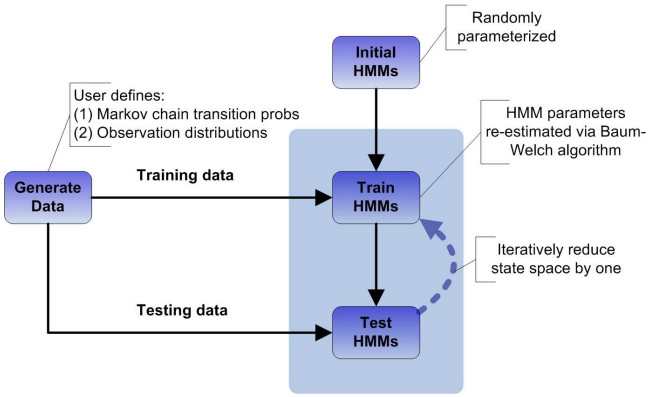

Figure 18. HMM model complexity experiment set-up. Experimental parameters, key functions with input/outputs listed, and looping constructs are shown.

Data generation follows a die-rolling paradigm. A sequence of die rolls produces a series of discrete observations. The stochastic data generation process uses 3 dice, each with 5-sides. A transition matrix of a Markov chain defines the probability of using a specific die with each die roll. An observation probability matrix defines the die bias. Two classes of data are generated. Each class has a different Markov chain transition matrix but uses the same observation distribution matrix. The data generation process forms training and testing sequences by choosing a die, rolling it, recording the result, and repeating the process to a user-defined sequence length termination. Experiment and data generation settings are shown in Table 5.

Training data are used to train discrete HMMs of varying complexity (i.e., number of hidden states). Trained HMMs are given test sequences from both classes of data. The class-specific discrete HMMs produce log-likelihoods when given test sequences. These likelihoods are compared and class assignment is made according to the most likely model.

Figures 19 and 20 show results from the discrete HMM complexity experiment. A marked jump in classification accuracy occurs when discrete HMM classifiers of order 3 (3 hidden states) are used. Classification performance remains relatively con-

Table 5. Experimental settings for two-class complexity experiment using discrete HMMs

parameter	class 1	class 2
Markov chain transition matrix	$\begin{bmatrix} .2 & .7 & .1 \\ .1 & .3 & .6 \\ .4 & .1 & .5 \end{bmatrix}$	$\begin{bmatrix} .1 & .1 & .8 \\ .7 & .1 & .2 \\ .1 & .6 & .3 \end{bmatrix}$
observation matrix	$\begin{bmatrix} .6 & .1 & .05 & .2 & .05 \\ .05 & .6 & .1 & .2 & .05 \\ .05 & .1 & .5 & .15 & .2 \end{bmatrix}$	
training records	100	
training seq length	30	
test records	1000	
test seq length	$\begin{bmatrix} 5 & 10 & 15 & 20 & 25 & 30 \end{bmatrix}$	
replications	5	

stant as model complexity increases. The Akaike and Bayesian information criterion (AIC and BIC) concur on the appropriate model complexity (minimum at 3 hidden states).

One measure of classifier performance is the Receiver Operating Characteristic (ROC) curve. ROC curves have been applied to many dichotomous decision problems [96]. Alsing [97] reviews ROC curve analysis in automatic target recognition research. A ROC curve is used to estimate classifier performance given test data. Typically, a ROC curve shows the range of false-positive/true-positive coordinates generated by varying a decision threshold from conservative to aggressive values. A conservative setting minimizes the number of false-positives (or false alarms) at the cost of reduced true-positive performance. An aggressive setting maximizes the true-positive performance at the cost of increased false-positives.

Figure 20 plots several ROC curves for the discrete HMM classifier given different sequence lengths. A longer sequence length means that the classifier has more observation data to consider before classification is made. The HMM classifier

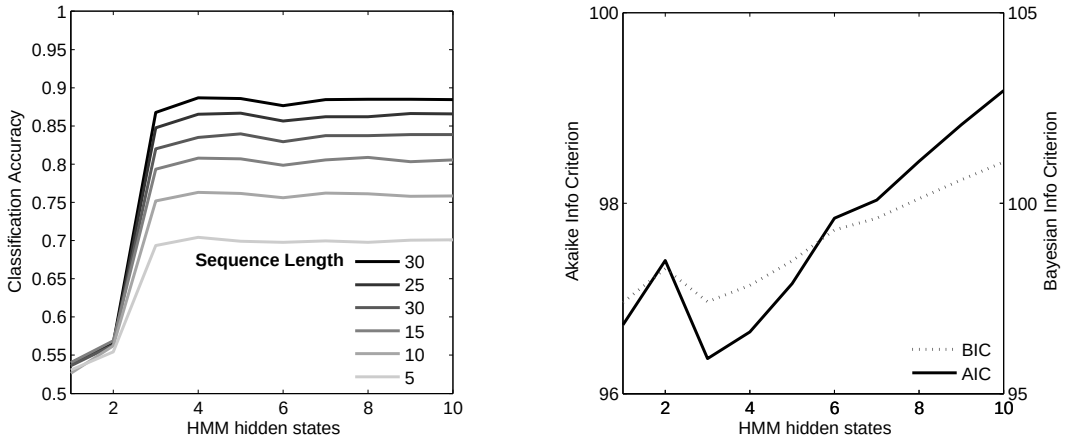

Figure 19. Discrete HMM complexity experiment results. On the left, classification accuracy results by sequence length across model complexity. On the right, AIC and BIC measures for model selection.

which produces the ROC curves shown in Fig. 20 is of order 3 (i.e., "best" model complexity).

The right-hand plot of Fig. 20 shows the distribution of observations in the training data for each class of data. A statistical classifier that did not account for the order of observation would have a difficult time distinguishing between the two classes.

3.3.2 Complexity in Continuous HMMs

Continuous hidden Markov models have a discrete state space and a continuous observation space. The following experiment examines measures of complexity in a Gaussian HMM-based classifier, i.e., the observation space related to each hidden state is distributed Gaussian. The experiment applies various measures of complexity to identify the most appropriate model given an ensemble of potential models. Data is generated from a stochastic model of known complexity. The data is then used to train and test a Gaussian HMM-based classifier. Based on classifier output, a model selection technique is applied. Then a comparison is made between the controlled,

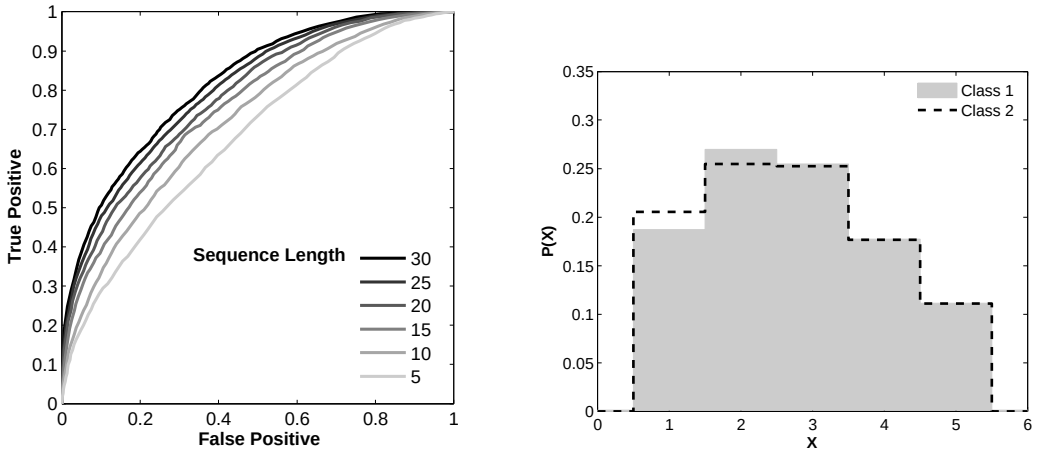

Figure 20. ROC curves for various sequence length settings. Discrete probability distribution for the 2 class problem.

user-defined data complexity and the suggested model complexity based on model outputs.

The experiment includes the following steps:

- Choose experiment parameters. Here the parameters of the stochastic model used to generate the data are specified.
- Generate training and testing data based on the parameter set
- Generate initial Gaussian HMMs of varying complexity
- Train the HMMs using a training data set
- Test the HMMs using a testing data set
- Reduce the HMM state space by one state and repeat the training/testing sequence

The output of an experiment is a mean log-likelihood achieved by averaging the log-likelihoods produced by the trained HMMs given each testing record. Thus as the number of testing records increases, the better the estimator (mean log-likelihood) for model performance.

Table 6. Experimental settings for two-class complexity experiment using continuous HMMs

parameter	class 1	class 2
Markov chain transition matrix	$\begin{bmatrix} .2 & .7 & .1 \\ .1 & .3 & .6 \\ .4 & .1 & .5 \end{bmatrix}$	$\begin{bmatrix} .3 & .7 & 0 \\ .4 & .0 & .6 \\ .3 & .4 & .3 \end{bmatrix}$
Gaussian observation matrix	$\begin{bmatrix} 1 & 2 & 5 \\ 1 & 2 & .5 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 5 \\ 1 & 2 & .5 \end{bmatrix}$
training records	100	
training seq length	40	
test records	1000	
test seq length	$\begin{bmatrix} 5 & 10 & 15 & 20 & 25 & 30 \end{bmatrix}$	
discretization	$\begin{bmatrix} 5 & 10 & 30 \end{bmatrix}$	
replications	5	

Data generation uses a Markov chain of 3 states to determine the observation distribution from which the next observation is drawn. Table 6 shows the experimental settings. For example, if the Markov chain is in state 1, the observation is randomly drawn from a Gaussian distribution with mean = 1 and variance = 1. Two classes of data are generated. Each class has a different Markov chain transition matrix but uses the same observation distribution matrix.

Training data are used to train Gaussian HMMs of varying complexity (i.e. number of hidden states). Trained HMMs are given test sequences from both classes of data. The class-specific HMMs produce log-likelihoods when given test sequences. These likelihoods are compared and class assignment is made according to the most likely model.

Discrete HMMs are trained using the same continuous data after quantizing using a k-means clustering algorithm. To compare performance of discrete versus Gaussian HMMs, several quantization levels are used (k = 5, 10, and 30). When $k = 5$ the discrete observation space has 5 symbols. Figure 21 shows classification

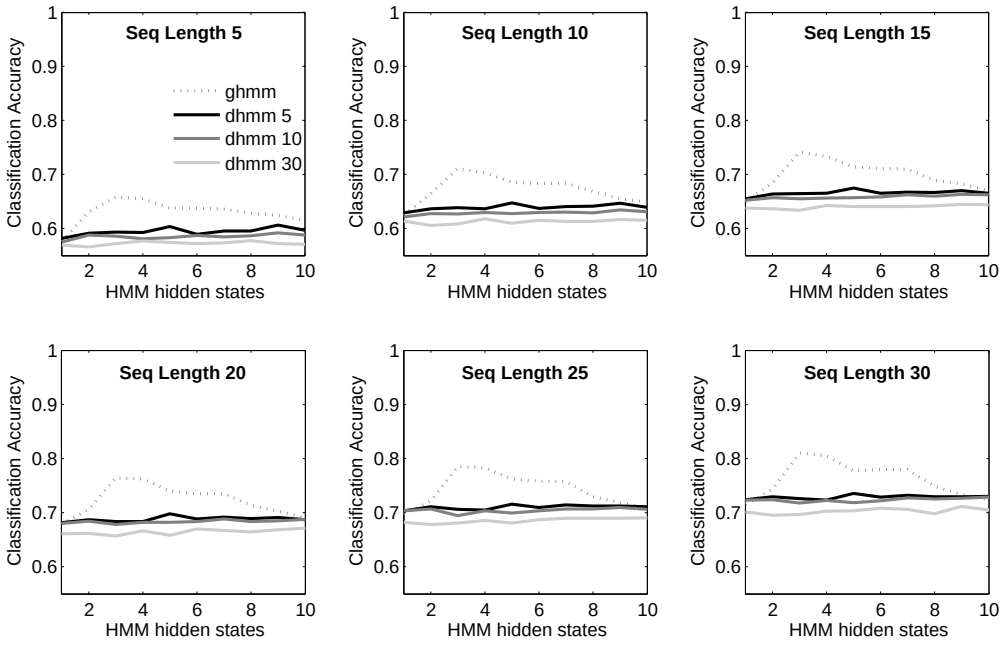

Figure 21. Classification performance of the Gaussian and discrete HMM classifiers at different sequence length settings. The discrete HMM classifier is broken down into three quantization levels: 5, 10, and 30.

accuracy as a function of model complexity at a series of sequence length settings. Notice the improved performance of the Gaussian HMM over the discrete HMM and the marked peak at HMMs of order 3.

Figures 22 and 23 show results from the Gaussian HMM complexity experiment. A marked jump in classification accuracy occurs with HMM classifiers of order 3 (3 hidden states). Classification performance decreases as model complexity increases. The Akaike and Bayesian information criterion (AIC and BIC) concur on the appropriate model complexity (minimum at 3 hidden states).

Figure 23 plots several ROC curves for the Gaussian HMM classifier given different sequence lengths. A longer sequence length means that the classifier has more observation data to consider before classification. The HMM classifier which produced the ROC curves shown in Fig. 23 is of order 3 (i.e., "best" model complexity).

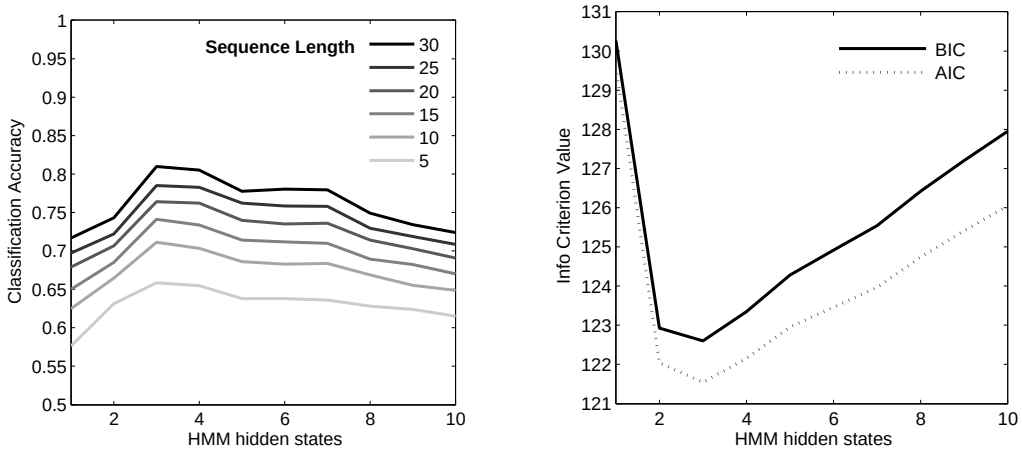

Figure 22. Continuous HMM complexity experimental results. On the left, classification accuracy results by sequence length across model complexity. On the right, AIC and BIC measures for model selection.

The right-hand plot of Fig. 23 shows the distribution of observations in the training data for each class of data. A statistical classifier that did not account for the order of observation would have a difficult time distinguishing between the two classes.

3.3.3 Multi-dimensional Gaussian Data

This section uses synthetic, multi-variate Gaussian data to show the utility of Gaussian HMMs in classifying sequenced, multi-variate data.

Table 7 lists the experimental parameters. Data is generated in a controlled manner using a specified number of states (5). Data in each state is generated from 3-dimensional random normal distributions. The mean and variance of each normal distribution depends on the state and data class.

Gaussian HMMs with 5 hidden states are trained using a sequence of samples (10) from each 3-dimensional Gaussian observation state. Thus each observation sequence is of length 50. Within an observation sequence the data is ordered by the distribution state. For example, the first 10 observations in the sequence are from

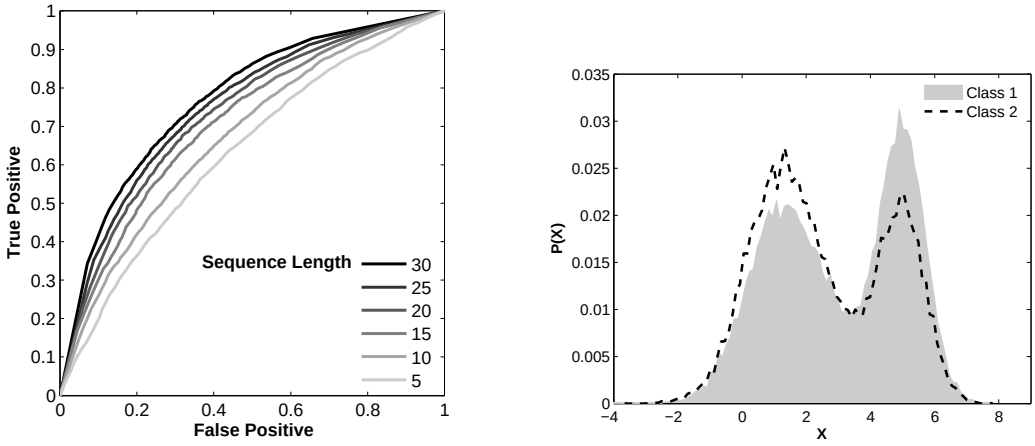

Figure 23. ROC curves for various sequence length settings. Continuous probability densities for the 2 class problem.

Table 7. Experimental settings for two-class complexity experiment using multivariate Gaussian HMMs

parameter	class 1	class 2
data mean by state	$\begin{bmatrix} 0 & 1.5 & 3 & 4.5 & 6 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} .1 & 1.6 & 3.1 & 4.6 & 6.1 \\ .1 & .1 & .1 & .1 & .1 \\ .1 & .1 & .1 & .1 & .1 \end{bmatrix}$
data std dev by state	$\begin{bmatrix} .4 & .4 & .4 & .4 & .4 \end{bmatrix}$	$\begin{bmatrix} .4 & .4 & .4 & .4 & .4 \end{bmatrix}$
samples per state	10	
training records	10	
test records	100	
mean separation settings	$\begin{bmatrix} .1 & .12 & .15 & .2 & .25 & .3 \end{bmatrix}$	
replications	5	

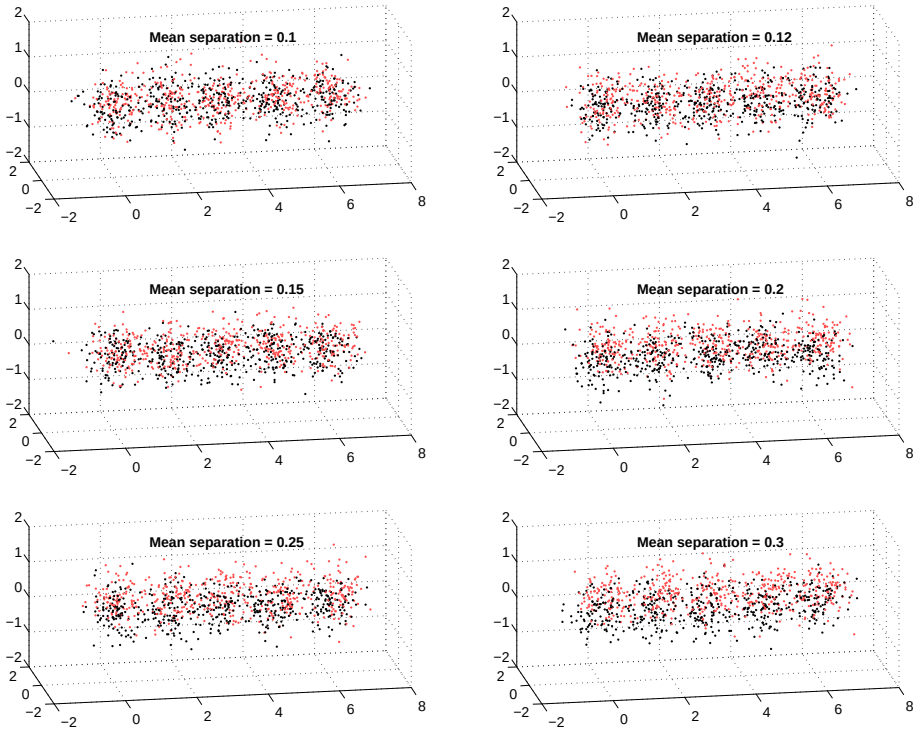

Figure 24. Two classes of data shown at six different mean separation settings.

state 1, the next 10 are from state 2, etc. In this fashion an ordering is forced on the data generation process. A sample of the two-class data is shown in Fig. 24.

Separate training and testing data are generated for each of the two data classes. The data classes are defined by the mean and standard deviation of their respective multi-variate Gaussian observation distributions. The experiment examines the ability of the Gaussian HMMs to distinguish between the two classes while the separation between the means is decreased from 0.3 to 0.1.

Figure 25 shows classifier performance using ROC curves at each of the mean separation settings. At the closest setting (0.1), the classifier narrowly outperforms the chance line (diagonal line). Perfect classification occurs with mean separation of 0.3. The results of this experiment point the way to implementation of a multi-variate Gaussian HMM in the application of CID using features from HRR signatures.

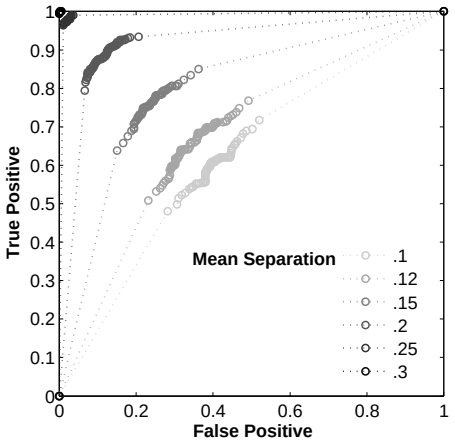

Figure 25. ROC curves at six different mean separation settings.

3.4 Development of HMM-based CID System

One goal of the research is the development of a time-series classifier operating on sequenced observations of targets. The hidden Markov model is the time-series classifier. In the following subsections data, design, and fusion decisions related to the implementation of an HMM-based classifier are described.

Figure 26 provides an experiment flowchart which shows the major processes used in the development of the HMM-based classifier. Feature data is derived from MSTAR SAR chips, and HMM classifiers are trained with class-specific data from a segregated training data set. The trained HMMs process test data, and a fuser combines the output from the class-specific HMMs and assigns class membership.

The following sections describe design options in the these areas: HRR-derived features, HMM state space structure, HMM observation distributions, HMM training, HMM testing, and fusion rule.

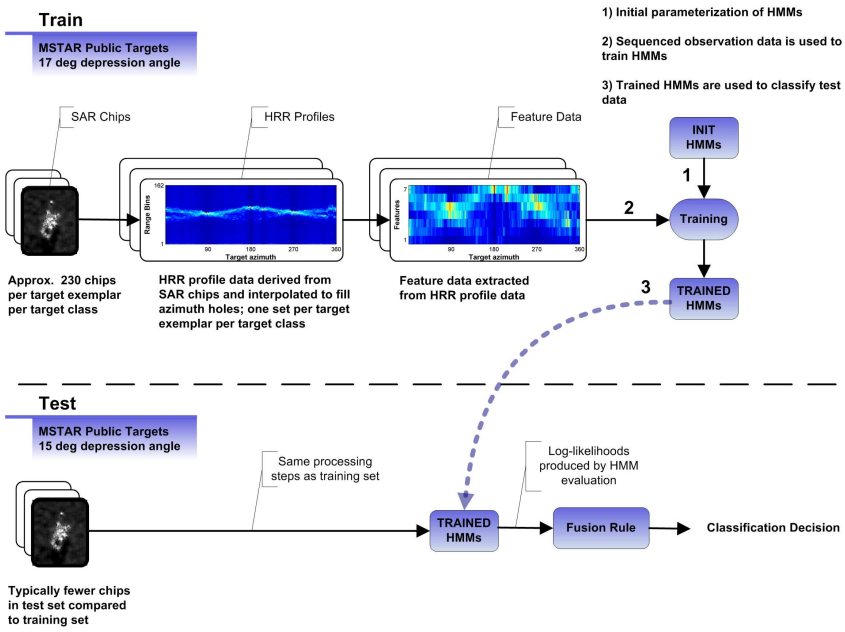

Figure 26. Set-up for a standard HMM experiment using features derived from MSTAR SAR chips and employing a fusion rule to combine outputs from multiple HMM classifiers.

3.4.1 Data and Features

A time-series classifier using data derived from targets imaged at a series of aspect angles may leverage aspect-dependent information in the effort to distinguish targets.

The classifier development described here uses data from the MSTAR program publicly-available data set [65], a subset of Collection 1 taken Sep 1995 at the Redstone Arsenal, Huntsville, AL by the Sandia National Laboratory STARLOS sensor (airborne), operating at X-band in one foot resolution spotlight mode. Three types of ground targets are in the target set: T-72 main battle tank, and BMP-2 and BTR-70 armored personnel carriers. Figure 27 shows photographs and example SAR images of the ground targets in the data collection.

The data is divided into two sets. The first is used for training and is collected at a sensor-to-target depression angle of 17 degrees. The training set holds approxi-

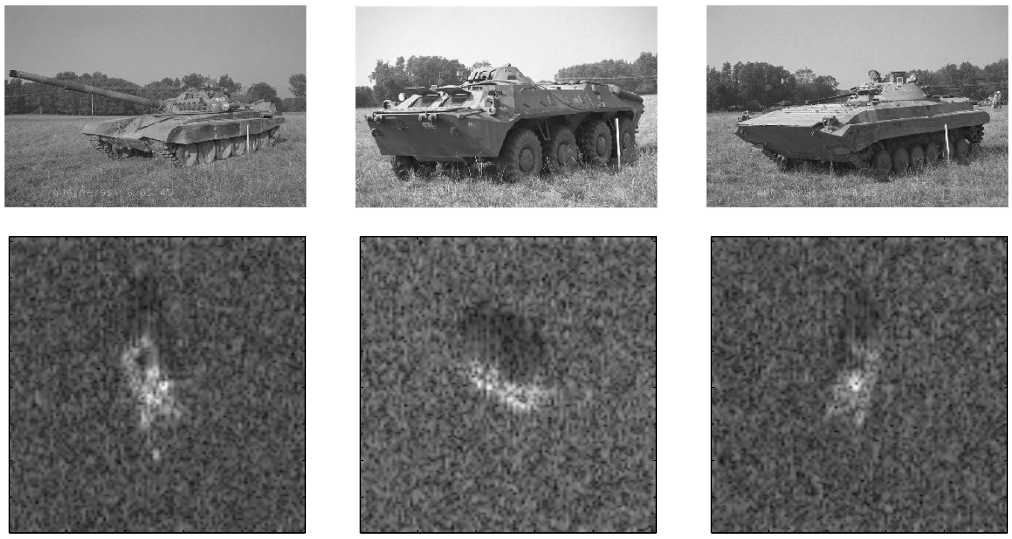

Figure 27. Target images and SAR chips of T-72, BTR-70, and BMP-2 vehicles.

mately 230 SAR chips of each target type with each chip representing a SAR image taken at a specific target aspect angle relative to the airborne sensor bore sight. The test set is collected at a 15 degree depression angle, presenting a signature different than the training set [98]. Approximately 195 SAR chips of each target type are in the test set.

The SAR chips are processed according to the steps of Section 2.2.6. A mean HRR signature is produced from each SAR chip. This signature, also called a profile, is a vector of length 162. Ordering the profiles by the sensor-target aspect angle creates a target HRR signature with respect to relative target azimuth. There are holes in the aspect data; approximately 230 chips cover 360 degrees of target azimuth. Figure 28 displays the available HRR profiles of each target (first column), the available profiles with missing data (second column), and interpolated profiles (third column). Profile data is linearly interpolated to 1 degree resolution using the available data, thus filling in the missing data and achieving a uniform spacing of observation data across target type.

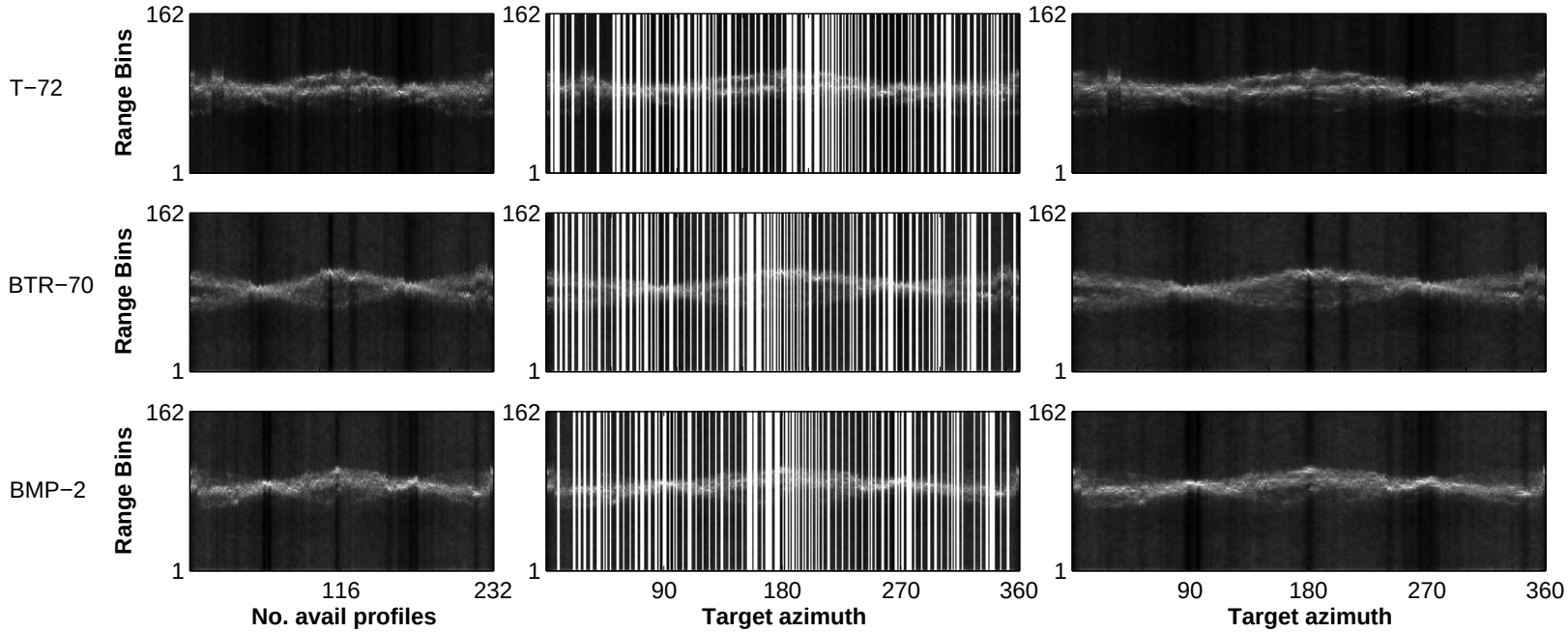

Figure 28. Available and interpolated HRR profiles for three MSTAR target types. The training data have a 17° depression angle.

Several methods are used to reduce the dimensionality of the 162-element HRR profile into manageable features. The first method is a simple maximum value rule applied to a series of adjacent bin ranges of the HRR profile. Each SAR chip is preprocessed (by the Sensors Directorate of AFRL) to place the target in the center of the chip. Figure 28 shows that most of the variability in the HRR signatures is confined to the middle region of the 162-element profiles. Further inspection showed that peaks of significant magnitude in HRR signatures for the three targets are located between range bins 62 and 100 of the 162-element profile. This range is divided into 7 range windows; 62-67, 68-73, 74-78, 79-83, 84-88, 89-94, and 95-100. The feature vector x is determined by the maximum values of the HRR profile within each of the seven range windows:

$$x_i^{(\max)} = \operatorname{argmax} p\left[w_i\right] \quad \text{for} \quad i = 1, 2, \ldots, 7 \tag{45}$$

where p is the HRR profile and w_i is the i^{th} range window of the HRR profile. A mean value feature rule is also used and is designated $x_i^{(\text{mean})}$.

The next feature set applies a discrete Fourier transform to the HRR profile:

$$x_i^{(\text{fft})} = \sum_{j=1}^{N} p(j)\omega_N^{(j-1)(i-1)} \quad \text{for} \quad i = 1, 2, \ldots, 6, \tag{46}$$

where $\omega_N = e^{(-2\pi i)/N}$ and $N = 162$. Most information is captured in the low frequencies of the transform, thus the feature vector retains only the first six values.

Another feature set is formed using principal component analysis (PCA). A translation of the middle portion of the HRR profiles (bins 62-100) across 360 degrees of aspect angle via principal component analysis reduces dimensionality from 39 to 10 when the first 10 principal components are retained. The component scores form the new feature space.

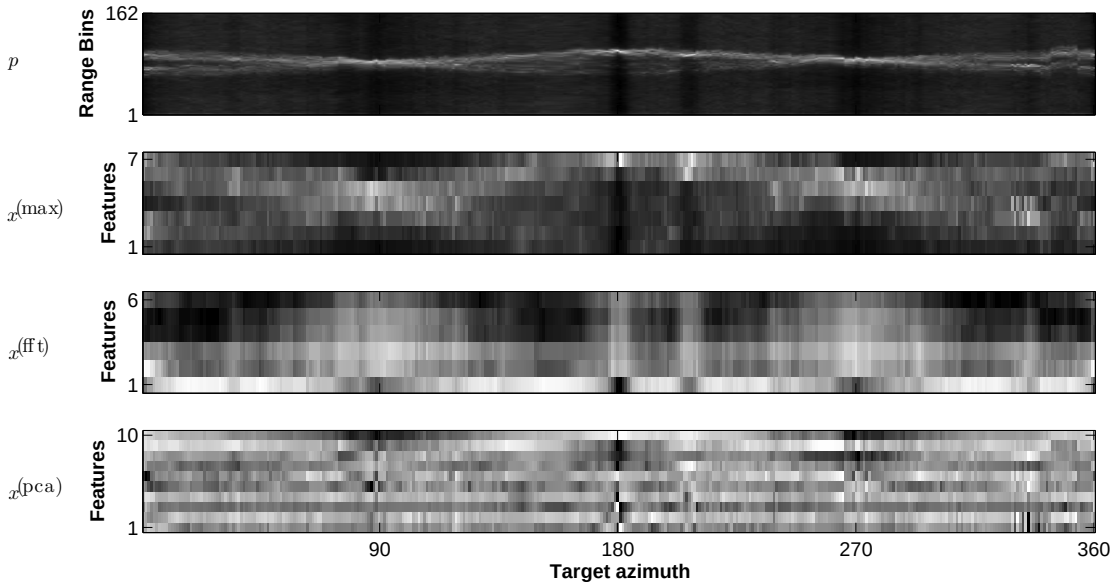

Figure 29. Full HRR profile (interpolated) and related feature sets for the BTR-70 target. Training data for a 17° depression angle are shown.

Given a 360 by 39 matrix, where each row is the middle portion of a full HRR profile at a given aspect angle, let p be the mean-corrected matrix such that $p_{ij} = p_{ij} - \mu_j$ where $\mu_j = (1/360) \sum_i p_{ij}$. Proceeding down the columns increases the aspect angle at which the HRR data are collected. Let C, a 39 by 39 matrix, be the normalized, sample variance-covariance matrix of p. Let A be the matrix of the eigenvectors associated with the ten largest eigenvalues of C. The component scores which form the new feature space result from multiplying the mean-corrected matrix p by A:

$$x^{(\mathrm{pca})} = p\,A. \tag{47}$$

Figure 29 shows the HRR-derived feature sets for the BTR-70 target. The full profiles p are interpolated to 1 degree resolution in aspect. The feature sets shown are: $x^{(\max)}$, the maximum value within 7 range windows; $x^{(\mathrm{fft})}$, the first 6 frequencies of the discrete Fourier transform; and $x^{(\mathrm{pca})}$, the component scores after translation using the first 10 principal components.

3.4.2 HMM Topology

A hidden Markov model λ is parameterized by the hidden Markov chain transition matrix A, an observation distribution matrix B, and an initial state probability vector π. The design of an HMM involves several decisions regarding topology. Of critical importance is the number of hidden states in the Markov chain, called the order of the HMM. Given S states, the transition matrix is $S \times S$. Thus, the number of parameters in the HMM increases non-linearly with S.

The state space may be fully-connected, in which case each state transitions to any other state in one step with probability greater than zero. Alternatively, the connectivity of the state space may be restricted. A specific instance is the "left-right" model, where a process in state i at time t is allowed only two options, either remain in state i at time $t+1$ or transition to state $i+1$ at time $t+1$. Thus the state transition matrix A has entries on the main and first diagonal with zeros elsewhere:

$$A = \begin{bmatrix} .3 & .7 & 0 & 0 & 0 \\ 0 & .2 & .8 & 0 & 0 \\ 0 & 0 & .4 & .6 & 0 \\ 0 & 0 & 0 & .5 & .5 \\ .4 & 0 & 0 & 0 & .6 \end{bmatrix}.$$

Another topology decision is modeling of the observation space. A discrete HMM employs a discrete observation space, called an alphabet, which consists of Q symbols. A discrete observation probability matrix defines the probability of producing a symbol given the state of the model. The matrix B has dimension $S \times Q$, and the number of parameters in the model grows linearly with Q.

A discrete HMM may be used to model continuous observation data, but the data must be quantized using some method, typically a k-means clustering, into

a discrete alphabet. Information is lost during the quantizing process, but model performance may not decrease substantially.

A Gaussian HMM assumes that the observation space is normally distributed, and B is no longer an observation distribution matrix in the sense of the discrete HMM. Instead, B contains the parameter pair μ and σ^2 for the Gaussian associated with each hidden state.

The feature sets considered in the following model development are multi-dimensional. For instance, an observation vector in the maximum value feature set $x^{(\max)}$ has dimension 7. The assumed Gaussian observation space is multi-dimensional, and a decision must be made to model the observations using seven 1-D HMMs, or one 7-D HMM, or some combination of lower-dimensioned HMMs.

The initial state distribution allows control of the initial state of the Markov chain. By setting π to a uniform distribution over all states, no prior information is inserted about the initial state.

Suppose, that a relationship must be forced between the hidden states and the observation sequence, e.g., for the aspect angle of the ground target in the SAR-derived observation data. As described above, the observations are ordered by aspect angle beginning at 1 degree and ending with 360 degrees. If the observations are assumed to be a function of the aspect angle of the target, i.e., when viewed from a certain aspect window, the observations are from a specific state in the hidden process. Thus, given an observation sequence that begins at a target aspect angle of 1 degree, the model can be forced to start in state 1 by setting the first element of π to 1 with zeros elsewhere.

3.4.3 Fusion Approaches

In a basic HMM-based classifier, one model is trained for each class of data. A test record of unknown class is evaluated by each class-specific HMM, producing

a log-likelihood, and class membership is assigned according to the greatest log-likelihood. If the test record is of dimension n and the classification system uses 1-D HMMs, then for each class of data n HMMs must be trained. For a three class problem such as that described above using 1-D HMMs operating on the maximum value feature set $x^{(\max)}$, the MCS consists of $3 \times 7 = 21$ HMMs. The output from this bank of models must be fused to assign a final class label.

Figure 30 schematically describes majority vote and mean log-likelihood fusion schemes. The MCS considers 1-D HMMs operating on two feature sets, where feature sets can originate from separate sensors. As noted above, when using 1-D HMMs, a model must be trained for every class of data and for every dimension of the observation feature vector, where 1-D observation sequences (step 1) are evaluated by the bank of trained HMMs (step 2) producing log-likelihoods (step 3).

Two methods are used to fuse within the feature set (i.e., produce a single sensor class label). A majority vote scheme tallies the winning votes for each class-specific model across the n dimensions of the feature set. Class membership is assigned to the class with the greatest number of votes. A mean log-likelihood scheme computes the mean output for each class across the n dimensions of the feature set. Class membership is assigned to the class with the largest mean log-likelihood. The process is repeated for the second feature set.

The same schemes can be used to fuse across the feature sets. If feature set 1 has dimension 7 and feature set 2 has dimension 6, then the voting scheme assigns class membership by tallying across 13 dimensions. Likewise, the mean log-likelihood scheme incorporates all 13 features before the final class assignment.

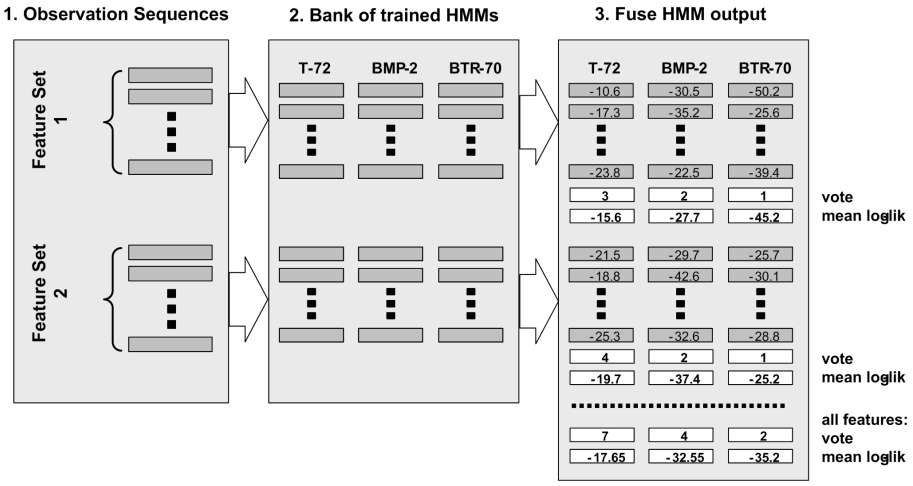

Figure 30. Fusion of multiple one-dimensional HMMs using a voting scheme and mean log-likelihoods.

3.4.4 Development Results

3.4.4.1 Discrete HMM

This section describes a discrete HMM-based MCS used to classify sequenced observations derived from MSTAR SAR data. The experiment described here is derived from Albrecht and Gustafson's conference paper [99]. The data consists of SAR chips of three ground targets (T-72, BTR-70, and BMP-2) collected from an airborne sensor. Each chip is a 2-D signature centered on a single target. Targets are not occluded and ground clutter is minimal (light vegetation).

Target SAR chips are processed into HRR signatures, and the signatures are then ordered by the relative sensor-target aspect angle. Features are extracted from the HRR signatures. Sequences of features, ordered by increasing apsect angle, form the observations used to train the discrete HMM-based MCS. Training data are collected at a depression angle of 17° and testing data are collected at 15°.

Two sets of features, representing information from two sensors to be fused later, are extracted from target HRR signatures. The first feature set, called "bin

feature set" is the maximum value within the 7 HRR range windows of Section 3.4.1, $x^{(\max)}$. The second feature set is the discrete Fourier transform of the HRR signature, called "FFT feature set," $x^{(\text{fft})}$.

The observation data are quantized in order to apply discrete HMMs. A linear quantization method is used to transform the continuous data into an alphabet with Q symbols. This method maps feature data into Q uniformly spaced intervals based on the minimum and maximum values of the training feature data. The state space of the discrete HMMs is fully-connected and consists of S states.

The experiment explores classification performance by varying the number of states in the discrete HMMs, i.e., $S = 2, 3, \ldots, 20$, the length of the observation sequence, and the method of fusing the component classifier outputs.

Figure 31 presents classification performance of the discrete HMM-based MCS using only the bin feature set. Seven HMMs per target type are employed in the MCS. The results shown in Fig. 31 reflect performance using HMMs with discrete observation alphabets of 10 symbols (i.e., $Q = 10$). Not shown are results at $Q = 30$, where performance dropped considerably below that seen with $Q = 10$.

Several trends are evident in the subplots of Fig. 31. First, as the observation sequence length increases classification performance increases. This result follows intuitively as the classifier is presented more information with a longer sequence length. Second, a general downward trend in performance coincides with increasing model complexity. Third, fusing the seven HMM outputs using the mean log-likelihood rule results in significantly improved performance over the majority vote rule at lower model complexity settings. Finally, the fusion rules perform better than their components acting independently.

Figure 32 presents classification performance of the discrete HMM-based MCS using only the FFT feature set. Six HMMs per target type are employed in the MCS. The results shown in Fig. 32 reflect performance using HMMs with discrete

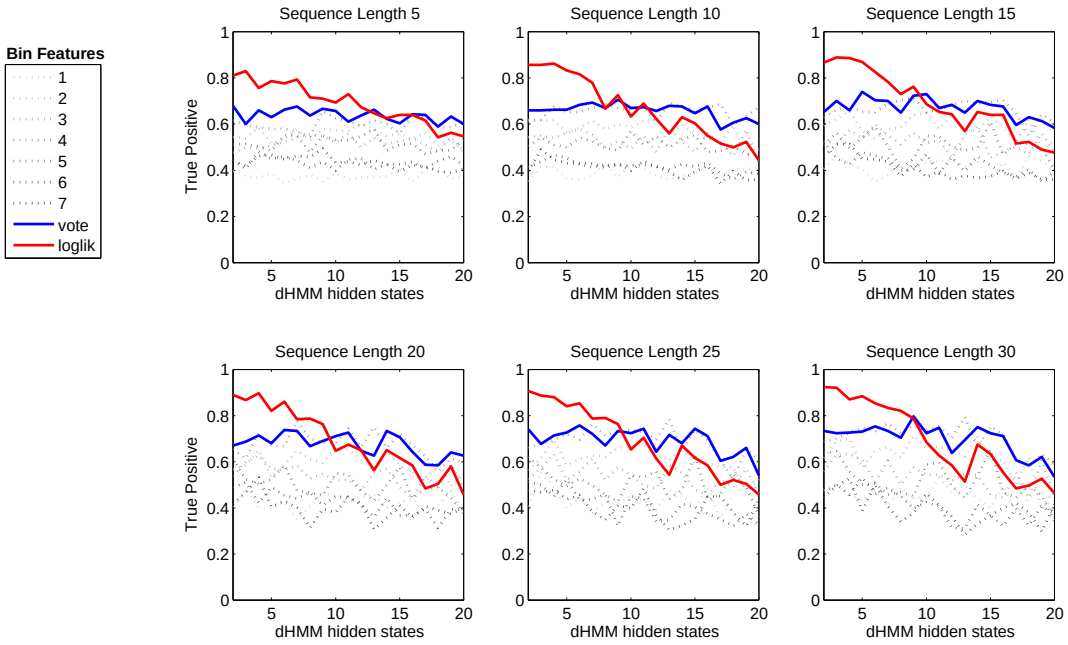

Figure 31. Fusion of multiple one-dimensional HMMs using a voting scheme and mean log-likelihoods operating on bin features.

observation alphabets of 10 symbols (i.e., $Q = 10$). Not shown are results at $Q = 30$, where performance dropped considerably below that seen with $Q = 10$.

Several trends are evident in the subplots of Fig. 32. First, compared to the bin feature set of Fig. 31, the FFT feature set performs poorly. There is insignificant improvement as sequence length increase, and classifier performance is relatively independent of model complexity. However, the fusion rules perform better than their components acting independently.

Figure 33 presents classification performance of the discrete HMM-based MCS using both feature sets. Thirteen HMMs per target type are employed in the MCS. The results shown in Fig. 33 reflect performance using HMMs with discrete observation alphabets of 10 symbols (i.e., $Q = 10$). Not shown are results at $Q = 30$, where performance dropped considerably below that seen with $Q = 10$.

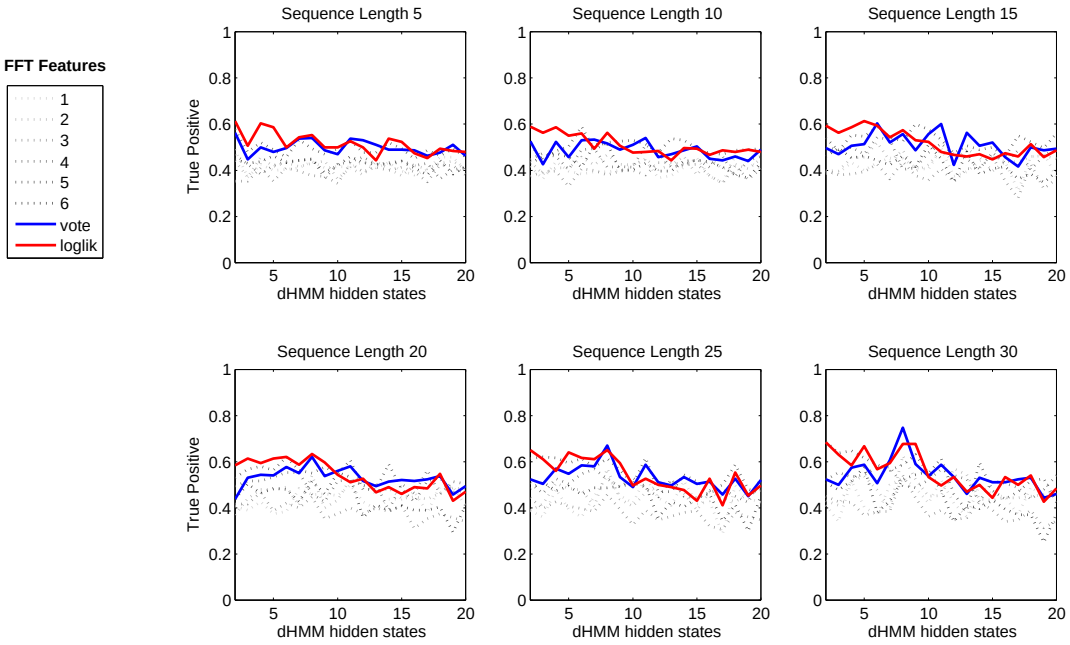

Figure 32. Fusion of multiple one-dimensional HMMs using a voting scheme and mean log-likelihoods operating on FFT features.

Fusing both feature sets improves performance over using either feature set alone. The trends in Fig. 31 can be seen in the fused performance. As the observation sequence length increases classification performance increases. A general downward trend in performance coincides with increasing model complexity. Fusing outputs with the mean log-likelihood rule results in significantly better performance than the majority vote rule at lower model complexity settings.

A random target classifier operating on 3 targets has a probability of correct selection (PCS) of 33%. The HMM-based classifier used in this experiment peaked at 94% PCS and typically operated between 70 and 80% at low model complexity and small alphabet size settings.

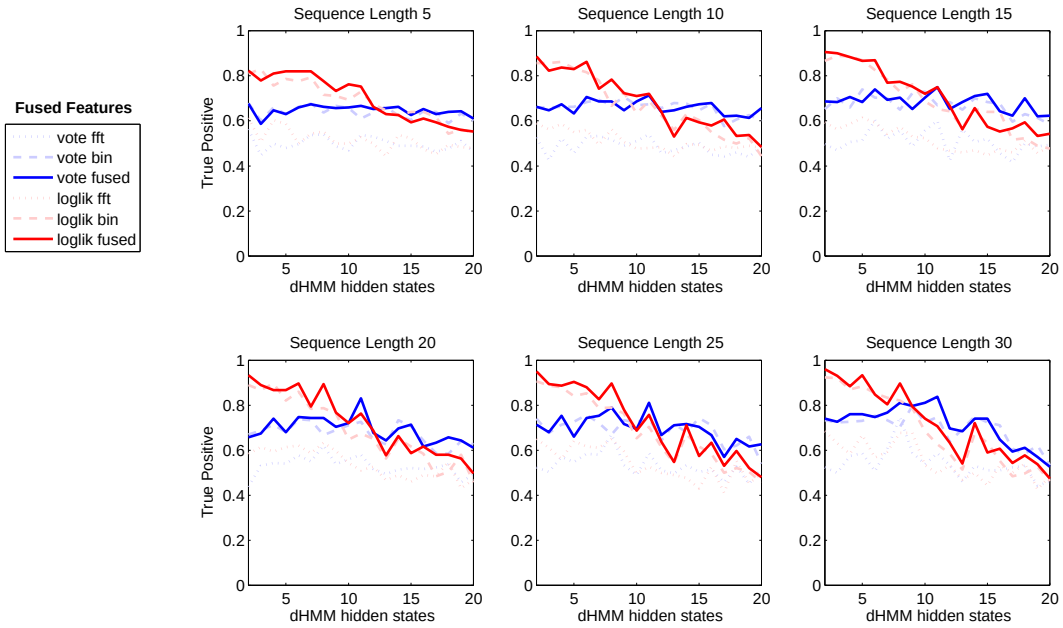

Figure 33. Fusion of multiple one-dimensional HMMs using a voting scheme and mean log-likelihoods across two feature sets.

3.4.4.2 Gaussian HMM

This section describes a Gaussian HMM-based MCS used to classify sequenced observations derived from MSTAR SAR data. The experiment described here extends the discrete research of the previous section [99] and is derived from Albrecht and Bauer's conference paper [100]. As in the discrete case, the data consist of SAR chips of three ground targets (T-72, BTR-70, and BMP-2) collected from an airborne sensor. Each chip is a 2-D signature centered on a single target. Targets are not occluded and ground clutter is minimal (light vegetation).

Two sets of features, representing information from two sensors to be fused later, are extracted from target HRR signatures. The first feature set, called "bin feature set" is the maximum value within 7 HRR range windows (see Section 3.4.1) $x^{(\max)}$. The second feature set is the discrete Fourier transform of the HRR signature, called "FFT feature set," $x^{(\text{fft})}$.

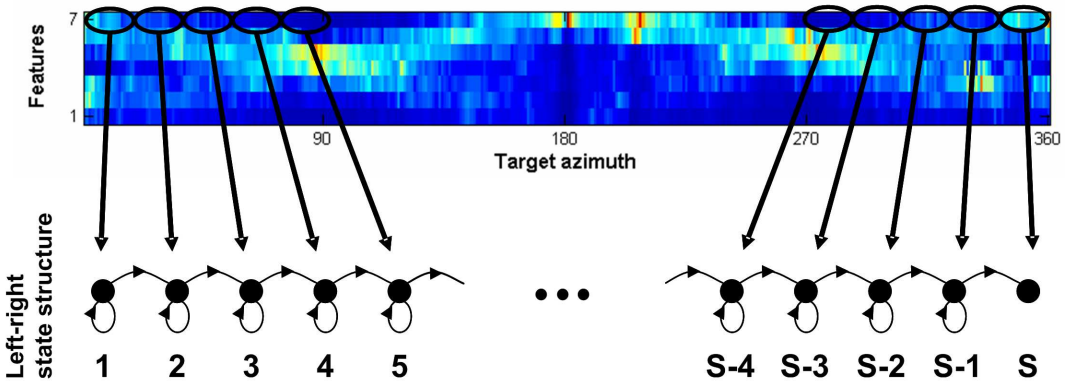

Figure 34. Observations in the feature space are linked to the observation distributions of the hidden states.

The state structure employed in the experiment described is a left-right model. The state transition matrix explicitly restricts transitions to two possibilities. First, the process remains in the same state (i.e., jumps to itself at the next time step). Second, the process transitions from a state to its adjacent neighbor. Using a left-right model may link observations to the ordered transition between states as shown in Fig. 34.

In a Gaussian HMM, observations from a given hidden state are distributed normally with parameters μ and σ^2, the mean and variance. The parameter re-estimation algorithm used to train the Gaussian HMM requires an initial parameter pair for each state distribution. Using the left-right model paradigm, each state is initially assumed to cover observations within a certain aspect window. For example, each state in a model containing 30 hidden states in the left-right state space initially covers a $360/30 = 12$ degree aspect window. The sample mean and variance of observations in the training data corresponding to the aspect window are used to initialize the Gaussian HMM state observation distribution parameters.

The experiment explores classification performance by varying the number of states in the Gaussian HMMs, $S = 10$, 20, 30, 40, 60, 72, and 90, the length of the observation sequence, and the method of fusing the component classifier outputs.

Figure 35 presents classification performance of the Gaussian HMM-based MCS. Because of the relationship between the hidden states and the target aspect angle, prior knowledge of the target pose can be used. In Fig. 35 no prior knowledge is used for classification.

Several trends are evident in the subplots of Fig. 35. First, as the observation sequence length increases classification performance increases. This result follows intuitively as the classifier is presented more information with a longer sequence length. Second, increased model complexity yields negligible performance improvement. Third, fusing the seven HMM outputs using the mean log-likelihood rule results in better performance than the majority vote rule across model complexity settings.

Figure 36 presents classification performance of the Gaussian HMM-based MCS with prior knowledge of target pose. The pose information is incorporated into the model by specifying the initial state when testing an observation sequence. Given a test sequence which begins with an observation of the target at 65° relative aspect angle and a Gaussian HMM with 30 hidden states (each state initialized to cover a 12° window), the initial state probability vector π is set to zeros everywhere except element 5, which is set to 1. Thus, the test sequence is evaluated with the hidden process beginning in state 5.

Incorporating prior target pose information significantly improves classification performance as seen in Fig. 36. Prior aspect knowledge also removes relative performance benefits between the two fusion methodologies.

3.4.4.3 Multi-dimensional Gaussian HMM

This section describes a multi-dimensional Gaussian HMM-based MCS used to classify sequenced observations derived from MSTAR SAR data. The experiment described here extends the one-dimensional Gaussian HMM research of the previous

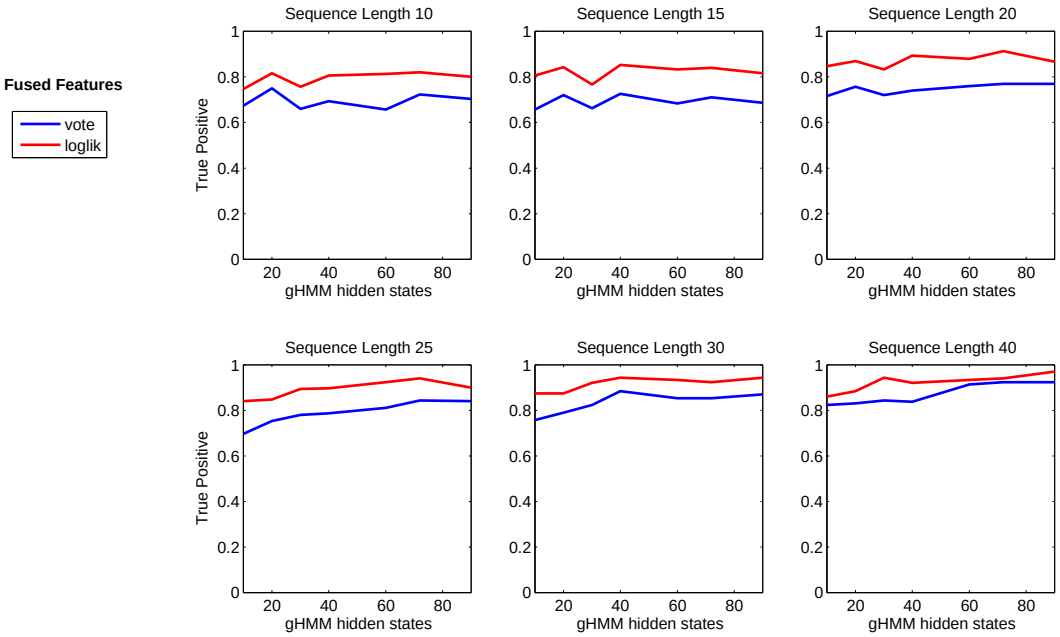

Figure 35. Fusion of multiple one-dimensional Gaussian HMMs using a voting scheme and mean log-likelihoods. No prior knowledge of target aspect angle is used.

section. As in the previous cases, the data consist of SAR chips of three ground targets (T-72, BTR-70, and BMP-2) collected from an airborne sensor. Each chip is a 2-D signature centered on a single target. Targets are not occluded and ground clutter is minimal (light vegetation).

Two sets of features, representing information from two sensors to be fused later, are extracted from target HRR signatures. The first feature set, called "bin feature set" is the maximum value within 7 HRR range windows (see Section 3.4.1) $x^{(\max)}$. The second feature set is the discrete Fourier transform of the HRR signature, called "FFT feature set," $x^{(\text{fft})}$. The state structure employed is a left-right model as in the previous section.

In a multi-dimensional Gaussian HMM, observations from a given hidden state are distributed multi-variate normally with parameters μ and $\mathbf{\Sigma}$, the mean vector and covariance matrix. The parameter re-estimation algorithm used to train the

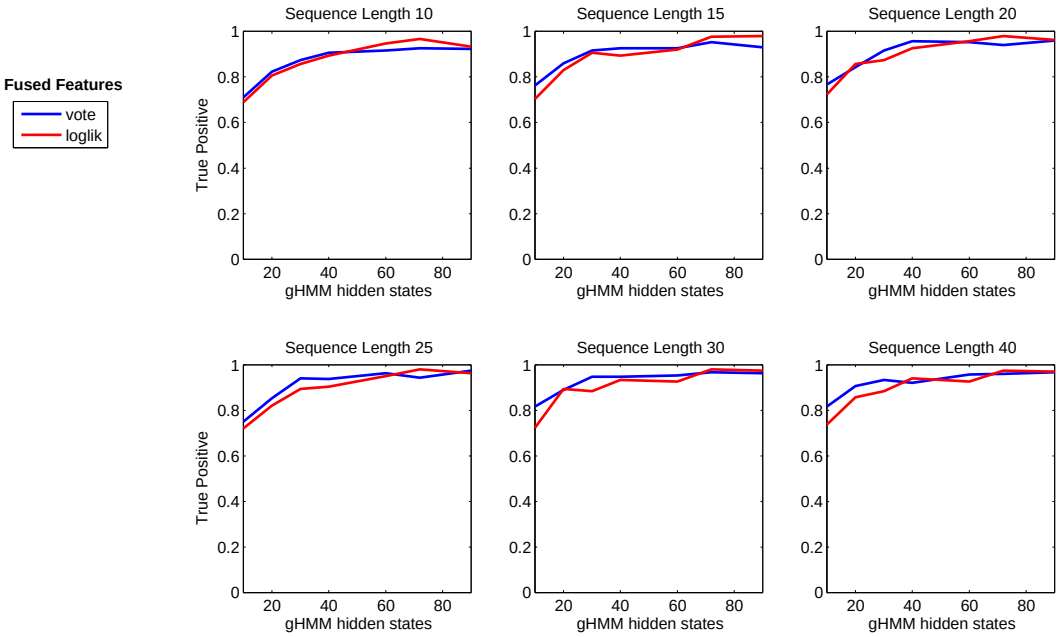

Figure 36. Fusion of multiple one-dimensional Gaussian HMMs using a voting scheme and mean log-likelihoods. Prior knowledge of target aspect angle as a function of the number of hidden states is used.

Gaussian HMM requires an initial parameter pair for each state distribution. Using the left-right model paradigm, each state is initially assumed to cover observations within a certain aspect window. For example, each state in a model containing 30 hidden states in the left-right state space initially covers a $360/30 = 12$ degree aspect window. The sample mean and covariance matrix of observations in the training data corresponding to the aspect window are used to initialize the Gaussian HMM state observation distribution parameters.

As seen in Fig. 37, the observation space is assumed to be multi-variate normal with dimension 7, which covers the feature space of the first feature set, $x^{(\max)}$. A second multi-dimensional Gaussian HMM is used to model the second feature set, $x^{(\text{fft})}$ with dimension 6. Thus, for each target only two HMMs are needed versus 13 models for the case of Sec. 3.4.4.2.

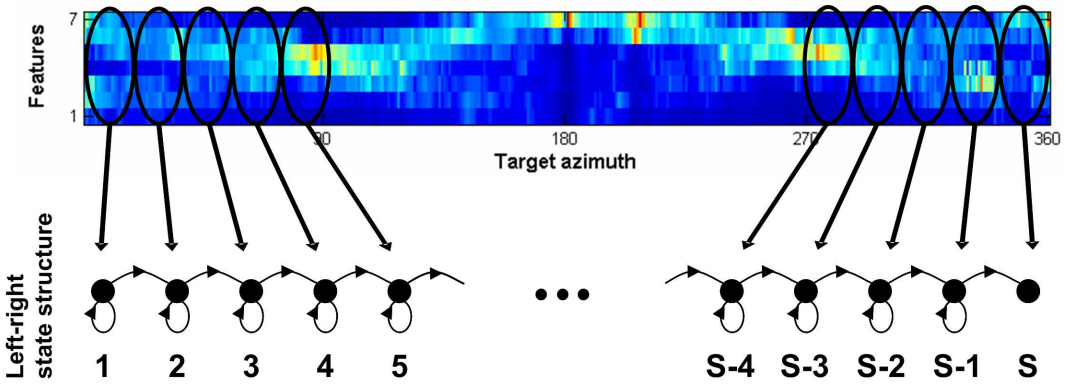

Figure 37. Multi-dimensional observations in the feature space are linked to the observation distributions of the hidden states.

With only two model outputs to combine, the majority vote fusion method is not used. Instead, only the mean log-likelihood method is used. The experiment explores classification performance by varying the number of states in the Gaussian HMMs, $S = 10$, 20, 30, 40, 60, 72, and 90, and the length of the observation sequence.

Figure 38 presents classification performance of the multi-dimensional Gaussian HMM-based MCS with no prior target aspect information. Several trends are evident in the subplots of Fig. 38. First, as the observation sequence length increases classification performance increases. This follows intuitively as the classifier is presented more information with a longer sequence length. Second, model performance decreases with increased model complexity.

Figure 39 presents classification performance of the multi-dimensional Gaussian HMM-based MCS with prior knowledge of the target pose. The pose information is incorporated into the model as described in Sec. 3.4.4.2. Incorporating prior target pose information improves classification performance. At the longest sequence length setting near perfect classification is achieved.

Figure 40 shows Akaike's information criterion (AIC) for the case of multi-dimensional Gaussian HMMs. Each dashed line represents AIC versus number of

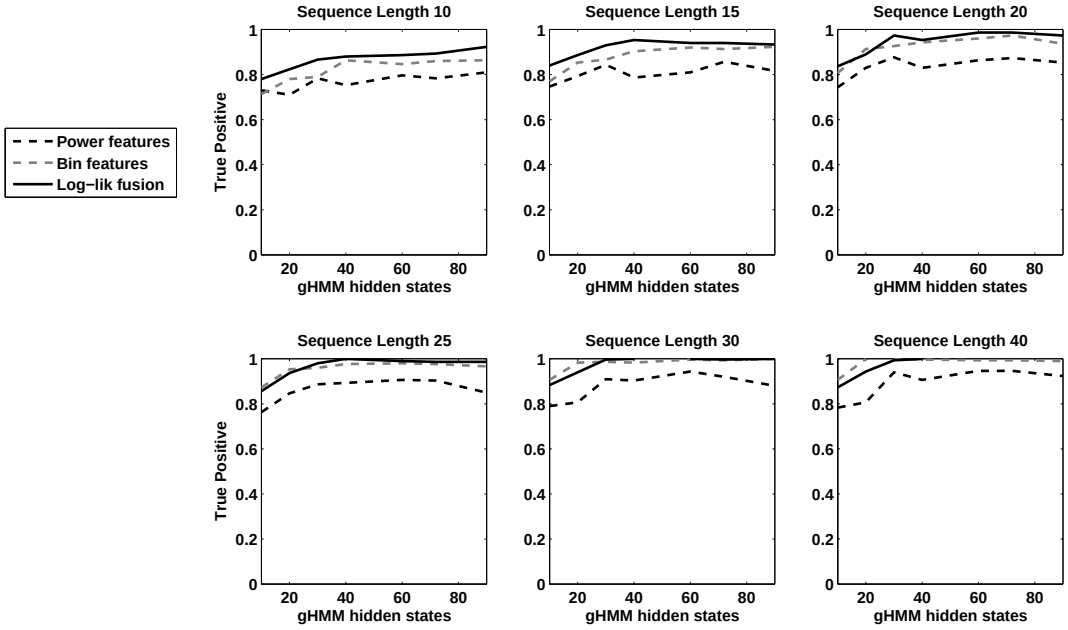

Figure 38. Fusion of multi-dimensional Gaussian HMMs using mean log-likelihoods. No prior knowledge of target aspect angle is used.

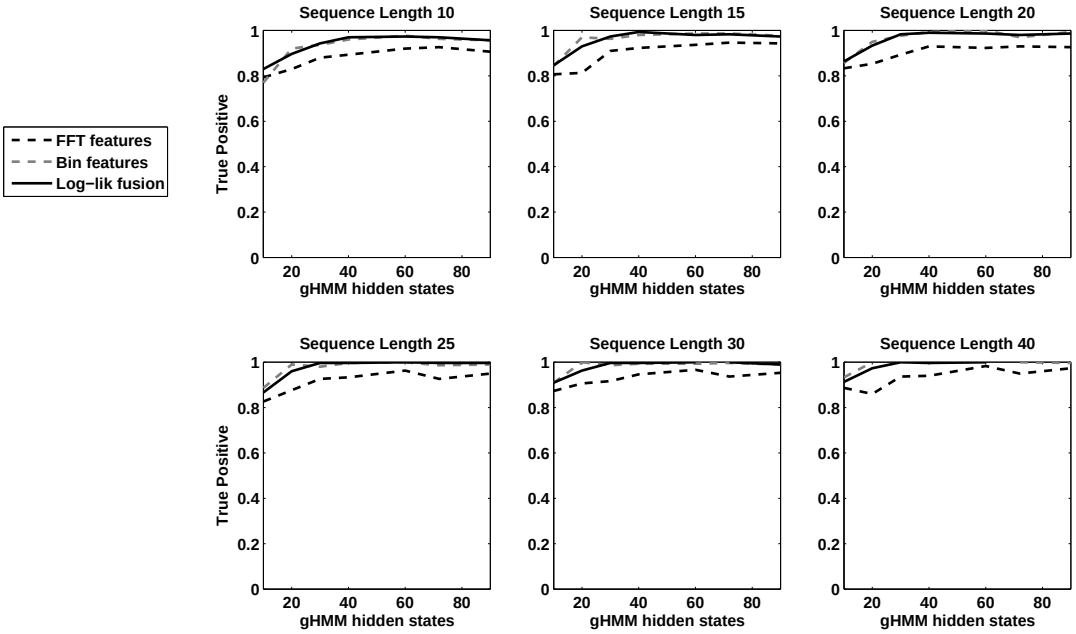

Figure 39. Fusion of multi-dimensional Gaussian HMMs using mean log-likelihoods. Prior knowledge of target aspect angle as a function of the number of hidden states is used.

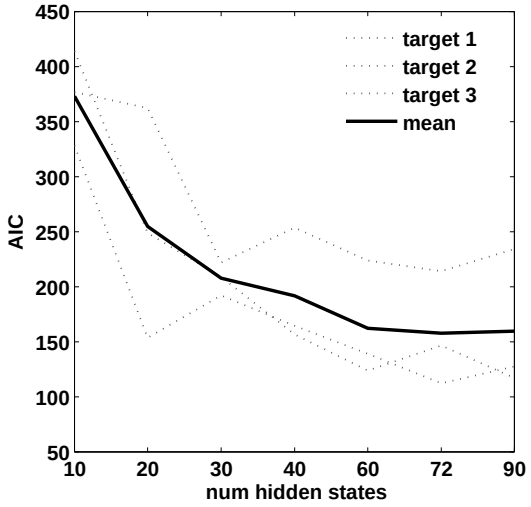

Figure 40. Akaike's information criterion versus number of hidden states for the multi-dimensional Gaussian HMM. Three dashed lines represent AIC for target-specific models. The solid line is the mean AIC across target models.

hidden states in a class specific model. Since the target set has three members, there are three AIC lines. The solid line is the mean AIC across all three target types.

As the number of hidden states increases, the amount of information lost (AIC) decreases, reaching a minimum at 72 hidden states. The AIC at 60 and 90 hidden states is approximately equal to that of 72 hidden states.

The AIC suggests that an appropriate level of model complexity given the training data used in the multi-dimensional Gaussian HMM experiment is either 60, 72, or 90 hidden states.

3.5 Summary

This chapter provides an introductory example of a discrete HMM applied in a genetic sequence classification experiment. In addition, it applies model complexity theory to the study of HMMs. Sections 3.3.1 and 3.3.2 apply AIC and BIC information theoretic measures to discrete and continuous HMMs in a controlled experiment

to identify appropriate model complexity. In these experiments, data are generated using Markov chains with 3 states. HMMs of varying complexity are trained and tested, and resulting AIC and BIC measures are calculated. In each case AIC and BIC concur that an HMM of order 3 is the best-suited model given the data.

Sections 3.4.4.1, 3.4.4.2, and 3.4.4.3 detail the development of discrete, 1-dimensional, and multi-dimensional Gaussian HMMs for a ATR classifier using sequenced SAR data as input. Performance measures and model topology suggest that a multi-dimensional Gaussian HMM with 60, 72, or 90 hidden states is most appropriate given SAR-based feature data as in Sec. 3.4.1.

4. CID Optimization Formulation

This chapter presents a combat identification (CID) optimization formulation that extends the CID framework proposed by Laine [15]. It begins by defining CID and automatic target recognition (ATR) related terms. Next, CID analysis using receiver operating characteristic (ROC) curves and confusion matrices is covered. Finally, Laine's framework is covered, and the extension to include out-of-library methodology is presented.

4.1 Definitions

The following terms related to research in the area of CID and ATR are defined prior to the presentation of the proposed extended framework.

ATD/R Automatic target detection and recognition refers to the process of detecting a region of interest (ROI) where a target may reside. The assumption in this research is that target detection has been accomplished, an ROI is established, and target recognition is the primary function.

ATR Automatic target recognition refers to the process of classifying objects in the ROI. In this research, ATR is performed with no human in-the-loop. Figure 41 shows a notional ATR system which incorporates two sensors and a fusion rule to combine sensor output prior to labeling the target.

CID Combat identification is the process of obtaining accurate characterizations of detected objects in the joint battlespace to the extent that high confidence and timely application of military options and weapons resources can occur [3]. An ATR system may be part of a CID system. A pilot's eyes may be part of a CID system.

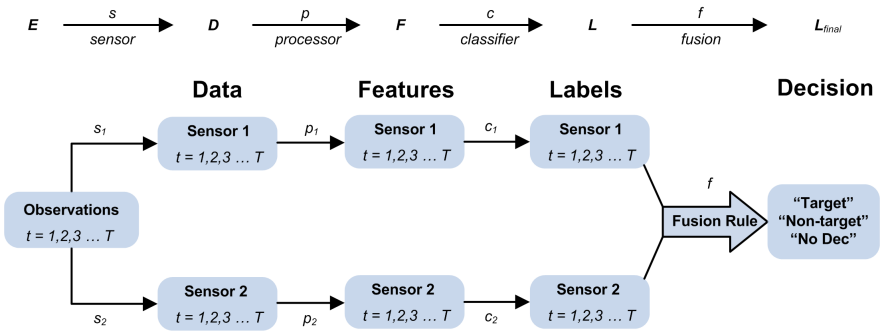

Figure 41. Notional ATR system with two sensors evaluating observations through time $t = T$.

Clutter Clutter encompasses the set of naturally occurring objects that degrade sensor performance. Examples of clutter include trees, rocks, and vegetation.

Confuser Confusers are man-made objects which a sensor may confuse with a true, in-library target. A decoy is an example of a confuser.

EOC Extended operating conditions are those variations of target presentation and environment which alter the sensed target signature from that of the training signature. Examples of EOCs include turret and barrel position of a tank, dense foliage versus sparse foliage, and depression angle from airborne sensor to ground target.

In-library refers to target types present in the classifier training set.

Label is the output of a classifier, or multi-classifier ATR system, when presented an ROI. Labels include "hostile," "friend," and "non-declare."

Out-of-library refers to target types not present in the classifier training set. The classifier has not been trained to recognize these targets.

ROI An ATR system is cued to a region of interest in order to classify the target residing in the ROI.

Target class refers to a grouping of similar target types. The grouping may depend on target intent (hostile, friendly, neutral), country of origin (U.S., NATO, or China) or vehicle type (tank, missile launcher, or truck).

Target type refers to classification based on high-fidelity physical properties of the target. Variants of the T-72 all fit within the T-72 target type. Another main battle tank, the M-1A1 and its variants form another target type.

4.2 ROC and confusion matrix analysis

Traditional ATR performance analysis uses ROC curves and confusion matrices to estimate system performance [97]. ROC curves relate classification performance by moving a threshold from conservative to aggressive settings. Typically, a ROC curve shows the trade-off between true-positive and false-positive performance as a function of a moving ROC threshold, $\theta \in [0, 1]$, from 0 to 1.

At each threshold setting true-positive and false-positive calculations are made based on class posterior probabilities output by the classifier. Given a threshold θ and two-class posterior probabilities, ppT (target) and ppF (friend), the classifier labels test records according to:

$$\text{label} = \begin{cases} \text{“target”} & \text{if } ppT \geq \theta \\ \text{“friend”} & \text{if } ppT < \theta \end{cases} \tag{48}$$

By comparing true class with classifier-assigned labels, true-positive and false-positive metrics are derived. Plotting the true-positive and false-positive pairings for each threshold setting produces a ROC curve.

Laine [15] introduces the idea of a ROC surface with the addition of a rejection option. A third performance measure, probability of declaration P_{dec}, is added to

the two already in use, probability of true-positive, P_{tp}, and probability of false-positive, P_{fp}. Here P_{dec} captures the number of records labeled "no declaration." The three measures are estimated as a function of the threshold θ such that

$$\hat{P}_{\text{tp}} = \hat{P}_{\text{tp}}(\theta)\,, \quad \hat{P}_{\text{fp}} = \hat{P}_{\text{fp}}(\theta)\,, \quad \text{and} \quad \hat{P}_{\text{dec}} = \hat{P}_{\text{dec}}(\theta) = 1 - \hat{P}_{\text{rej}}(\theta), \tag{49}$$

where $\hat{P}_{\text{rej}}$ is the estimated probability of labeling "non-declaration."

The ROC surface s is produced by varying θ over its range Θ:

$$s = s(\theta) = \left\{ \left(\hat{P}_{\text{tp}}(\theta)\,, \hat{P}_{\text{fp}}(\theta)\,, \hat{P}_{\text{dec}}(\theta) \right) | \, \theta \in \Theta \right\}, \tag{50}$$

where the threshold θ now defines a rejection region. The center of the rejection region is defined by θ_{ROC}, and the rejection region half-width is given by θ_{REJ}. Thus, the bounds on the rejection region are $(\theta_{\text{ROC}} - \theta_{\text{REJ}}\,,\, \theta_{\text{ROC}} + \theta_{\text{REJ}})$.

Labeling with a rejection option follows:

$$\text{label} = \begin{cases} \text{"target"} & \text{for } \; ppT > \theta_{\text{ROC}} + \theta_{\text{REJ}} \\ \text{"friend"} & \text{for } \; ppT < \theta_{\text{ROC}} - \theta_{\text{REJ}} \\ \text{"non-declare"} & \text{for } \; \theta_{\text{ROC}} - \theta_{\text{REJ}} \leq ppT \leq \theta_{\text{ROC}} + \theta_{\text{REJ}} \end{cases}. \tag{51}$$

Figure 42 depicts the labeling process given a rejection region. Two distributions of classifier-produced posterior probabilities are given. Records of true target class have higher posterior probabilities while true friend class have lower posterior probabilities. The two distributions overlap, creating classification errors given a decision boundary. By inserting a rejection region, the classifier declares only those records with high likelihood of class membership [89]. Classification errors are reduced at the expense of fewer declarations.

A ROC surface plots ROC curves across a third dimension which measures classifier declaration performance. Declaration performance is a function of the width

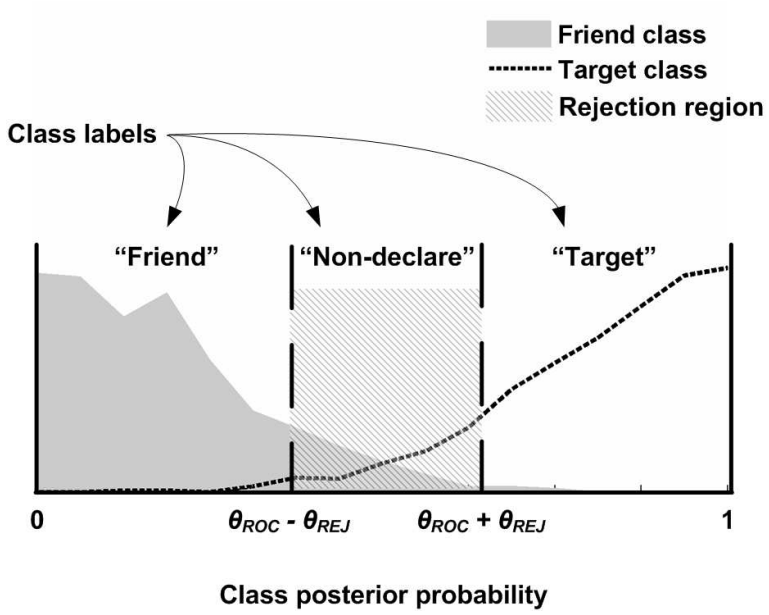

Figure 42. Rejection region based on ROC and rejection thresholds applied to class posterior probability.

of the rejection region; the wider the region, the more "non-declaration" labels, resulting in a lower declaration rate. By varying the θ_{ROC} and θ_{REJ} from conservative to aggressive settings, a ROC surface is produced. Figure 43 provides an example ROC surface. The plot shows decreased performance as declaration rate increases.

Analysis of CID system performance with confusion matrices yields a table of classifier labeling versus truth given a set of test data and thresholds (θ_{ROC} and θ_{REJ}). Figure 44 is a confusion matrix for a system with four classifier labels ("enemy," "friend," "neutral," and "non-declaration") and three true target classes. Each matrix entry represents test record labeling conditioned on true target class. For example, the first row shows the number of true-enemy records labeled "Enemy," "Friend," "Neutral," and "Non-declare," respectively. Reading horizontally indicates how well the classifier identifies true-enemy records. A common horizontal metric is the probability of true-positive, proportioned to the number of true-enemy records labeled "Enemy" given that a declaration is made.

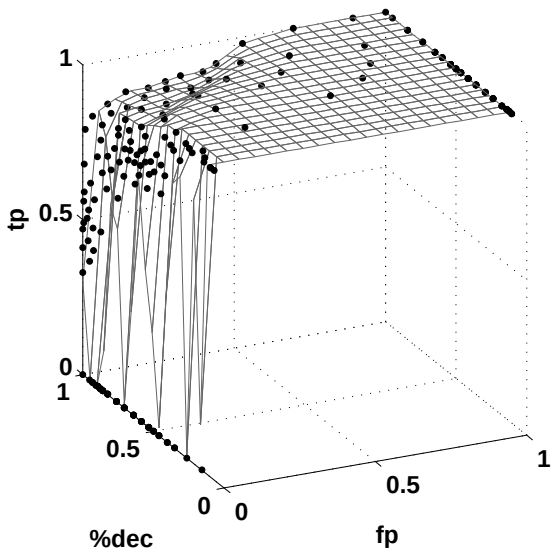

Figure 43. Family of ROC curves measuring true-positive and false-positive performance versus percentage of records declared. Points are experiment measurements.

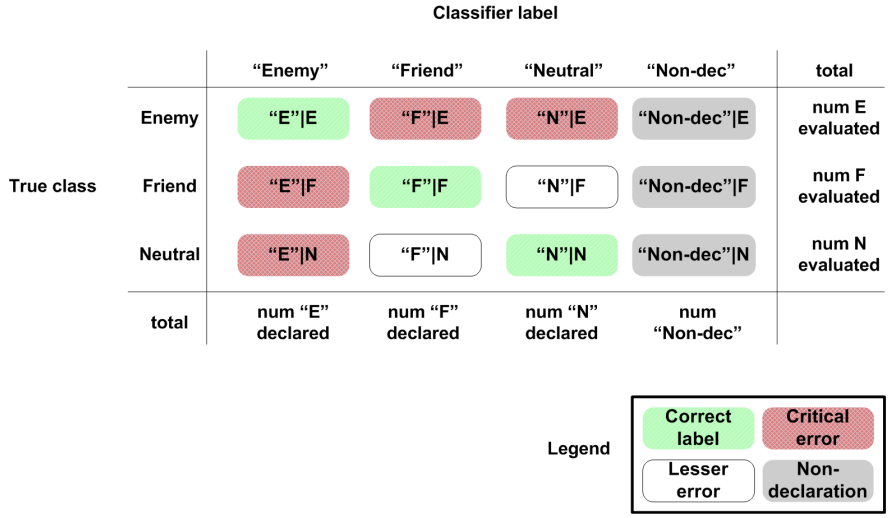

Figure 44. Confusion matrix with FEN classes and non-declaration option.

The columns of the confusion matrix indicate the true class of the records given a specific label. For example, the second column shows the respective number of true-enemy, friend, and neutral class records labeled "Friend" by the classifier. Reading vertically indicates classifier accuracy when applying the "Friend" label. One vertical metric is the fratricide rate; the number of "Enemy" labels applied to true-friend records given that a declaration is made.

Figure 44 uses hatching to distinguish between performance measures. The entries on the main diagonal reflect correct labeling of each true class of target. Critical errors includes both mislabeling a true-enemy as "Friend" or "Neutral," and applying an "Enemy" label to a true-friend or neutral. Lesser errors have little impact on warfighter decisions and include the cross-labeling of friend and neutral records. If friends and neutrals are treated in the same fashion, then cross-labeling errors are inconsequential.

Figure 45 shows a confusion matrix with two types of hostile targets and introduces non-critical errors as another performance measure. The number of true target classes remains the same by merging the friend and neutral classes due to the low impact of cross-labeling error. The enemy class is sub-divided into target-of-the-day (TOD) and other-hostile (OH) classes. This subdivision facilitates analysis of non-critical errors, which occur when incorrect hostile targets are engaged or when a weapons-target mismatch occurs. In either case, a suboptimal employment of resources occurs without loss of friendly/neutral life.

Figure 46 expands the number of true target classes to four with inclusion of an out-of-library class. Test records belonging to the out-of-library class are of target types not included in the classifier training set. Critical errors in Fig. 45 are similarly defined for Fig. 46. Non-critical errors expand to include mislabeling of true out-of-library targets as "Target-of-the-day" or "Other hostile," and labeling as "Out-of-library" those target of true-target-of-the-day or other-hostile class.

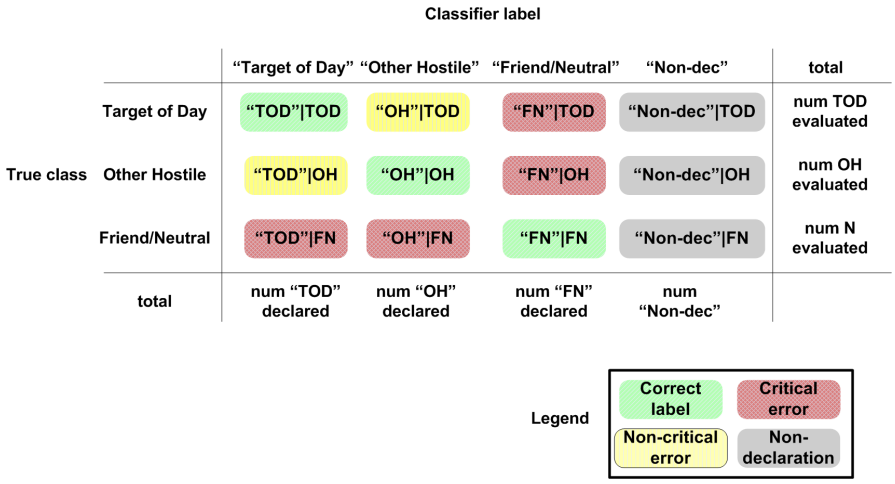

Figure 45. Confusion matrix with multiple hostile classes and non-declaration option.

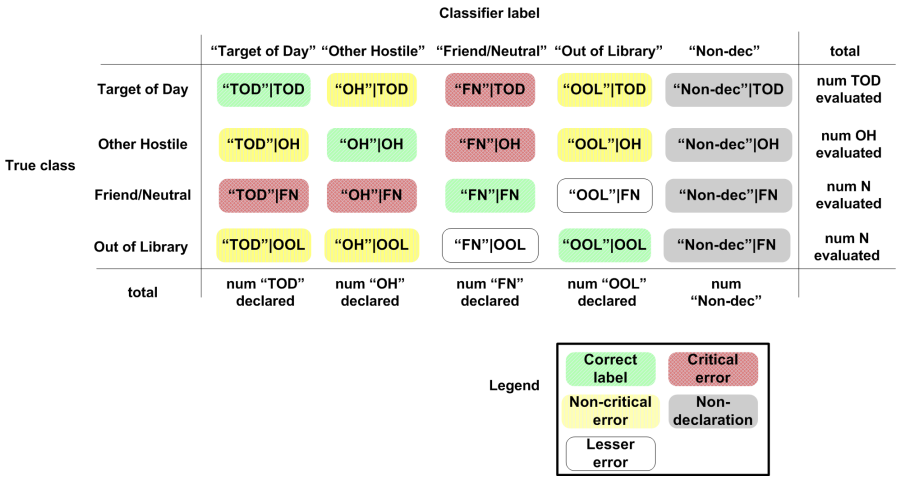

Figure 46. Confusion matrix with multiple hostile classes, out-of-library records, and non-declaration option.

4.3 Extended mathematical programming CID optimization formulation

Laine's optimization framework [15] uses a mathematical programming (MP) formulation to optimize CID systems without reference to fixed error costs. Nonlinear optimization of the decision space across classifier label mappings as a function of variable threshold settings provides a flexible objective function/constraint set pairing to suit warfighter preferences. For example, one strategy might be to maximize classifier true-positives while constraining the system to a maximum error rate and a minimum declaration rate.

Table 8 gives an initial MP CID formulation which includes an out-of-library performance measure. The initial formulation seeks to maximize the true-positive rate of the CID system subject to several constraints. The true-positive rate (TPR) is a measure of true-positives as a function of time or number of observations. The motivation is to capture the benefit of additional observations when classifying time-series data. Intuitively, a classifier performs better when given more (discriminatory) information, and TPR seeks to quantify the performance benefit of additional observations.

4.3.1 Decision variables

Decision variables used in the MP framework are organized into three groups: choice of fusion rule, choice of sensors, and choice of thresholds associated with the sensors and fusion rule.

Using Laine's notation [15], F_i is an indicator variable that describes use of the i^{th} fusion rule. Since the CID system under investigation fuses multiple sensors with one fusion rule, only one of f fusion rules may be selected in the optimal arrangement. Thus, $F_i = 1$ if the i^{th} fusion rule is chosen and all other entries are set to zero.

Table 8. Initial MP Formulation of CID Optimization Framework

Objective function	**MP formulation**	**Impact**
Maximize true positive rate	max TPR(x)	maximize number of true positives per look
Constraints		
Critical error	$E_{\text{crit}} < 0.02$	upper bound on critical error performance measure
Non-critical error	$E_{\text{ncrit}} < 0.05$	upper bound on non-critical error performance measure
True positive	$P_{\text{tp}} > 0.9$	lower bound on true positive performance measure
Declarations	$P_{\text{dec}} > 0.7$	lower bound on declaration performance measure
Out-of-library	$P_{\text{ool}} > 0.6$	lower bound on out-of-library performance measure

The fusion rule employs the output from an ensemble of sensors, where S_j is an indicator variable taking a value of 1 if sensor S_j is employed in the fusion scheme and 0 if not. Constraints may be employed to limit the selection to a certain number of sensors. For instance, the fused MCS must use at least one but not more than three sensors.

The third group of decision variables are the thresholds related to the fusion rule and component sensors. Using the index i to refer to the fusion method and j to refer to the component sensor, θ^{ij} is the threshold related to a specific CID system decision using fusion rule i and sensor j. Example thresholds are the ROC threshold θ^{0j}_{ROC} and rejection threshold θ^{0j}_{REJ}, which together define the rejection region at the classifier level for classifier j ($i = 0$ indicates that the threshold is not used at the fusion level).

4.3.2 Performance Measures

Given two competing CID systems, their performance is compared using estimates of true performance. The following sections develop these estimated performance measures (foregoing the estimator symbol ˆ).

4.3.2.1 True-positive

Horizontal analysis of confusion matrix entries produces performance estimates of class labels given true class. Some flexibility exists in using true-positive as a performance measure because the user identifies which class is the sought after target class. For this example, target classes are defined in Fig. 46 as TOD, OH, FN, and OOL with hostile targets TOD and OH as the target classes.

Thus an estimate for true-positive performance is the number of true-hostile records labeled "Hostile" divided by the total number of true-hostile records:

$$\begin{aligned} P_{\text{tp}} &= P(\text{“TOD”} \cup \text{“OH”} | \text{TOD} \cup \text{OH}) \\ &= \frac{\text{num}(\text{“TOD”} | \text{TOD} + \text{“TOD”} | \text{OH} + \text{“OH”} | \text{TOD} + \text{“OH”} | \text{OH})}{\text{num}(\text{TOD eval} + \text{OH eval})}. \end{aligned} \quad (52)$$

Further refinement of the performance measure may restrict the calculation to records on which a declaration is made by the classifier. This refinement accounts for the added rejection option and resultant "non-declare" label. Here the calculation is

$$\begin{aligned} P_{\text{tp}} &= P(\text{“TOD”} \cup \text{“OH”} | (\text{TOD} \cup \text{OH}) \cap \text{declaration}) \\ &= \frac{\text{num}(\text{“TOD”} | \text{TOD} + \text{“TOD”} | \text{OH} + \text{“OH”} | \text{TOD} + \text{“OH”} | \text{OH})}{\text{num}(\text{TOD declared} + \text{OH declared})}. \end{aligned} \quad (53)$$

4.3.2.2 Critical error

Critical error calculation reverses the order of conditioning seen in the true-positive calculation; instead of finding the probability of correct label given a true class, critical error finds the probability of true class membership given a label. Critical error calculation involves vertical analysis of the confusion matrix.

Using Fig. 46, critical error is

$$P(E_{\text{crit}}) = P\left(\left(\begin{array}{c} P(\text{“TOD”} \cap \text{FN}) \cup P(\text{“OH”} \cap \text{FN}) \cup \\ P(\text{“FN”} \cap \text{TOD}) \cup P(\text{“FN”} \cap \text{OH}) \end{array} \right) \mid \text{declaration} \right). \quad (54)$$

Simplification of Eq. 54 makes use of Bayes' rule, and depends on class prevalence as defined by class prior probabilities. Let $P(\text{TOD})$, $P(\text{OH})$, $P(\text{FN})$, and $P(\text{OOL})$ be the prior probabilities of the four true target classes, and let $P(\text{“TOD”})$, $P(\text{“OH”})$, $P(\text{“FN”})$, $P(\text{“OOL”})$, $P(\text{“Non-declare”})$ be the unconditional system label

probabilities. Then

$$P(\text{TOD}) + P(\text{OH}) + P(\text{FN}) + P(\text{OOL}) = 1 \quad (55)$$

$$P(\text{“TOD”}) + P(\text{“OH”}) + P(\text{“FN”}) + P(\text{“OOL”}) + P(\text{“Non-declare”}) = 1. \quad (56)$$

The simplified $P(E_{\text{crit}})$ calculation is then

$$P(E_{\text{crit}}) = \frac{\left(\begin{array}{c} P(\text{“TOD”}|\text{FN})P(\text{FN}) + P(\text{“OH”}|\text{FN})P(\text{FN}) + \\ P(\text{“FN”}|\text{TOD})P(\text{TOD}) + P(\text{“FN”}|\text{OH})P(\text{OH}) \end{array} \right)}{1 - P(\text{“Non-declare”})}, \quad (57)$$

where $P(\text{“Non-declare”})$ is determined by the sum of the class-specific probability of non-declaration:

$$\begin{aligned} P(\text{“Non-declare”}|\text{TOD})P(\text{TOD}) &+ P(\text{“Non-declare”}|\text{OH})P(\text{OH}) \\ &+ P(\text{“Non-declare”}|\text{FN})P(\text{FN}) \\ &+ P(\text{“Non-declare”}|\text{OOL})P(\text{OOL}). \end{aligned} \quad (58)$$

4.3.2.3 Non-critical error

As with critical error, non-critical error calculation reverses the order of conditioning in the true-positive calculation. Non-critical error calculation involves vertical analysis of the confusion matrix.

Some flexibility exists in choosing which classification errors constitute non-critical errors. For this example, non-critical errors consider only cross-labeled hostile targets (i.e., TOD labeled “OH” and OH labeled “TOD”), which is a reduced non-critical error set from that shown in Fig. 46. Adjusting the non-critical error set requires relatively simple modification to the following calculations:

$$P(E_{\text{ncrit}}) = P\left((P(\text{“TOD”}|\text{OH}) \cup P(\text{“OH”}|\text{TOD})\ )\,|\ \text{declaration}\right), \tag{59}$$

which simplifies to a non-critical error calculation that incorporates prior class probabilities, i.e.,

$$P(E_{\text{ncrit}}) = \frac{P(\text{“TOD”}|\text{OH})P(\text{OH}) + P(\text{“OH”}|\text{TOD})P(\text{TOD})}{1 - P(\text{“Non-declare”})}, \tag{60}$$

where $P(\text{“Non-declare”})$ is determined by the sum of the class-specific probability of non-declaration (see Eq. 75).

4.3.2.4 Declaration

The declaration performance measure captures the percentage of test records which the CID system labels with one of the true class labels. The complementary measure is the non-declaration performance measure. It tabulates the number of records labeled “Non-declare” by the system:

$$\begin{aligned} P_{\text{dec}} &= 1 - P(\text{“Non-declare”}) \\ &= 1 - \begin{pmatrix} P(\text{“Non-declare”}|\text{TOD})P(\text{TOD}) + P(\text{“Non-declare”}|\text{OH})P(\text{OH}) \\ +P(\text{“Non-declare”}|\text{FN})P(\text{FN}) \\ +P(\text{“Non-declare”}|\text{OOL})P(\text{OOL}) \end{pmatrix}. \end{aligned} \tag{61}$$

4.3.2.5 Out-of-library

The out-of-library performance measure is a true-positive labeling of “OOL” given an OOL record using horizontal analysis of confusion matrix entries. For this example, target classes are defined as in Fig. 46 (TOD, OH, FN, and OOL).

Thus, the estimate for the out-of-library performance measure is the number of true-OOL records labeled "OOL" divided by the total number of true-OOL records evaluated:

$$\begin{aligned} P_{\text{ool}} &= P(\text{“OOL”}|\text{OOL}) \\ &= \frac{\text{num}(\text{“OOL”}|\text{OOL})}{\text{num}(\text{OOL eval})}. \end{aligned} \tag{62}$$

Further refinement of the performance measure may restrict the calculation to records on which a declaration is made by the classifier. This refinement accounts for the added rejection option and resultant "non-declare" label. Thus the calculation is

$$\begin{aligned} P_{\text{ool}} &= P(\text{“OOL”}|\text{OOL}) \\ &= \frac{\text{num}(\text{“OOL”}|\text{OOL})}{\text{num}(\text{OOL declared})}. \end{aligned} \tag{63}$$

The out-of-library labeling methodology is separate from the labeling methodology for in-library classes. Figure 47 shows a notional implementation of the out-of-library labeling methodology. Focusing on a single sensor, observations of a region of interest are made through time and passed to a feature processor to extract features, which are the basis for classification. The classifier produces a 10-dimensional class posterior probability vector. Based on feature observations, this vector is the classifier's best guess at class membership. The vector is 10-dimensional because the classifier has been trained to recognize 10 in-library classes.

The proposed in-library/out-of-library discriminator takes the 10-dimensional class posterior probability vector as input and produces an 11-dimensional class posterior probability vector. The 11-D vector adds a posterior probability for the out-of-library (**OOL**) class as a function of the 10-D in-class vector.

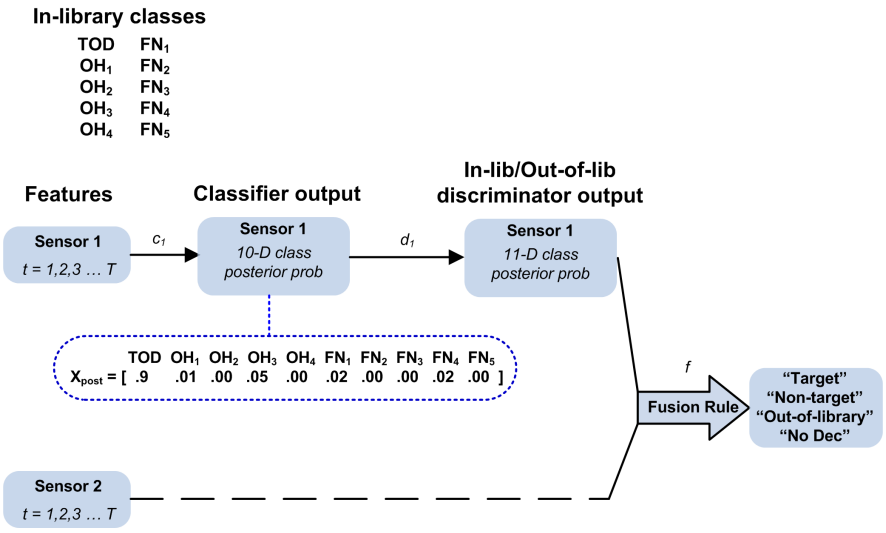

Figure 47. Out-of-library discriminator added to a two-sensor notional ATR system evaluating observations through time $t = T$. Given ten in-library classes, classifier outputs are 10-dimensional class posterior probabilities. The out-of-library discriminator assigns an 11th posterior as a function of the 10 in-library posteriors.

Given the 10-D in-class posterior probability vector

$$\mathbf{x}_{\text{post}} = [pp_{\text{TOD}} \quad pp_{\text{OH1}} \quad pp_{\text{OH2}} \quad \cdots \quad pp_{\text{FN5}}],$$

the discrimination function sorts the posteriors in descending order producing $\mathbf{x}_{\text{ord}}$. Assuming that the classifier identifies in-library targets well, a small subset of the class posteriors are significantly larger than the remaining posteriors. In-library/out-of-library discrimination results from a threshold setting based on the sum of a subset of ordered posteriors.

Two threshold parameters are chosen through a nearly blind sub-optimization routine. The parameters are the number of ordered (largest to smallest) posteriors over which to sum $\theta^{(1)}_{\text{OOL}}$, and the threshold against which the sum is compared $\theta^{(2)}_{\text{OOL}}$. The parameter values are chosen to ensure a minimum discrimination performance given in-library and out-of-library records, hence the nearly blind description. For

example, the sub-optimization routine may determine a threshold $\theta^{(2)}_{\text{OOL}}$ based on the sum of the second through sixth ordered posteriors, $\theta^{(1)}_{\text{OOL}} = 6$. Thus given $\mathbf{x}_{\text{ord}}$ for a sample test record, the discriminator compares

$$\mathbf{x}_{\text{ool}} = \sum_{i=2}^{\theta^{(1)}_{\text{OOL}}} \mathbf{x}_{\text{ord}}(i),$$

i.e., the sum of the second through sixth ordered posteriors for the test record with the threshold $\theta^{(2)}_{\text{OOL}}$, and it assigns the OOL posterior as a function of distance from the threshold

$$pp_{\text{OOL}} = \begin{cases} 0 & \text{if } \mathbf{x}_{\text{ool}} < \theta^{(2)}_{\text{OOL}} \\ f(\mathbf{x}_{\text{ool}} - \theta^{(2)}_{\text{OOL}}) & \text{if } \mathbf{x}_{\text{ool}} \geq \theta^{(2)}_{\text{OOL}} \end{cases}.$$

If $\mathbf{x}_{\text{ool}} < \theta^{(2)}_{\text{OOL}}$, then the record is considered an in-library class and the posterior probability for OOL is set to zero. If $\mathbf{x}_{\text{ool}} \geq \theta^{(2)}_{\text{OOL}}$, then the record is an out-of-library record and the posterior probability for OOL is set to a monotonically-increasing function of the distance from the threshold:

$$f(d) = \frac{2}{1 + e^{-10d}},$$

where $d = \mathbf{x}_{\text{ool}} - \theta^{(2)}_{\text{OOL}}$. Since $\mathbf{x}_{\text{ool}} \in [0, 9/10]$ and $\theta^{(2)}_{\text{OOL}} \in [0, 1]$, $d \in [0, 9/10]$ and f maps d to $[1, 1.999]$, where pp_{OOL} is concatenated to the end of the 10-element estimated posterior vector $\mathbf{x}_{\text{post}}$ and normalized to produce the estimated 11-element posterior probability vector.

4.3.3 Formulation

Laine [15] lets $\mathbf{x}$ be a vector of decision variables defined in the MP formulation. Some decision variables such as the fusion indicator variable are discrete, while others are continuous. The MP formulation seeks to find the optimal $\mathbf{x}$ in the space of the

discrete and continuous decision variables given an objection function and limiting constraints.

The structure of the MP formulation is flexible and can adapt to various objective functions and constraints per the goals of the warfighter or CID system analyst. What follows are example formulations given the previous discussion.

Objective Function

$$\max_{x \in X} \text{TPR}(x) = \frac{P_{\text{tp}}(x)}{\text{num looks}} \qquad \text{maximize true-positive rate} \tag{64}$$

Subject to:

Warfighter constraints

$$\begin{aligned}
E_{\text{crit}} &< \Pi_1 && \text{upper bound on critical errors} \\
E_{\text{ncrit}} &< \Pi_2 && \text{upper bound on non-critical errors} \\
P_{\text{tp}} &> \Pi_3 && \text{lower bound on true-positive performance} \\
P_{\text{dec}} &> \Pi_4 && \text{lower bound on declaration performance} \\
P_{\text{ool}} &> \Pi_5 && \text{lower bound on out-of-library performance}
\end{aligned}$$

Fusion rule constraint

$$\sum_{i=1}^{f} F_i = 1 \qquad \text{select a single fusion rule}$$

$$\text{where} \quad F_i = \begin{cases} 1 & \text{if } i\text{th fusion rule used} \\ 0 & \text{otherwise} \end{cases}$$

Sensor selection constraint

$$\sum_{j=1}^{s} S_j \leq s \qquad \text{select from available sensors}$$

$$\sum_{j=1}^{s} S_j \geq 1 \qquad \text{select at least one sensor}$$

$$\text{where} \quad S_j = \begin{cases} 1 & \text{if } j\text{th sensor used} \\ 0 & \text{otherwise} \end{cases}$$

Threshold constraints

$\theta^{ij} \geq 0$	lower threshold constraint
$\theta^{ij} \leq 1$	upper threshold constraint
	where θ^{ij} is the decision threshold associated with fusion rule i and sensor j. The decision threshold may be θ_{ROC} or θ_{REJ}.

Laine shows how budgetary constraints could be developed by applying a cost function to the research and development, procurement, and maintenance of fusion systems and sensors. Physical constraints, such as weight, space, and electromagnetic spectrum are also possible but not considered here.

5. Application of extended CID framework

In this chapter, the extended CID framework is exercised in a classification experiment using DCS radar data. This experiment competes an HMM-based classifier against a template-based classifier across a variety of experimental settings.

The chapter has the following sections: an introduction to the experiment, a description of the experiment data, classifier definitions, the experimental methodology, optimization framework, and experimental results. The results section is further expanded to include post-optimality analysis with the implementation of a designed experiment.

5.1 Introduction

The goal of this chapter is to apply the extended CID optimization framework in an experiment using observation data of ground targets collected from an airborne sensor. Two different classifiers compete within the framework across a variety of experimental settings. Among these settings are:

- warfighter constraints such as minimum critical error rate
- threshold settings for the classifiers (ROC and rejection region thresholds)
- fusion methodology (no fusion, mean fusion rule, neural network fusion, and boolean fusion)
- level of sensor independence
- observation sequence length

Figure 48 provides an overview of the ATR system. A region of interest is observed in time by two sensors. These sensors may be co-located on the same platform or located on separate platforms. The sensor data is processed into features which are then used to classify a target into one of four classes: target-of-the-day

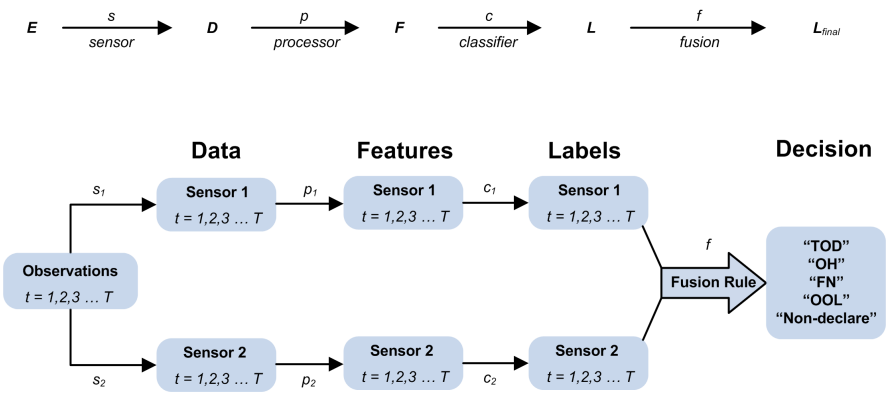

Figure 48. Overview of ATR system with two sensors sending observations through time $t = T$ to two classifiers whose outputs are fused into one of five labels: Target-of-the-day (TOD), Other hostile (OH), Friend/Neutral (FN), Out-of-library (OOL), or Non-declaration.

(TOD), other hostile (OH), friend/neutral (FN), or out-of-library (OOL). Should the classifier not have enough confidence (determined by thresholds) to label a target as belonging to one of the four classes, it applies a non-declare label.

5.2 Data description

The data set used in this experiment was collected May 2004 at Eglin Air Force Base, Florida. The AFRL Sensor Data Management System released the data in response to a data request through the website `https://www.sdms.afrl.af.mil`. The collection used a General Dynamics DCS X-band synthetic aperture (SAR) radar operating in spotlight mode aboard a medium-sized, twin-engined Convair 580. The radar bandwidth was 640 MHz with a peak transmit power of 4 kW. The DCS radar imagery was collected at a resolution of 1.0 ft in two channels; HH-polarization and VV-polarization. All targets were stationary and imaged in an open area without concealment using a spotlight mode, and SAR chips of individual targets were extracted from full spotlight scenes. The SAR chips used in this experiment had 256 x 256 pixels.

Table 9. DCS Collection target list by class with description and experiment labels.

Group	Type	Target description	Tracks	Wheels	Gun	Label
Hostile	SCUD	Single Large Missile	N	8	N	TOD
	SMERCH	MLRS Scud Confuser	N	8	N	OH1
	SA-6 Radar	Similar to SA-6 TEL	Y	0	N	OH2
	T-72	Main Battle Tank	Y	0	Y	OH3
	SA-6 TEL	3 Medium SAMs	Y	0	N	OH4
Friend and Neutral	Zil-131	Medium Budget Truck	N	4	N	FN1
	HMMWV	Jeep like SUV	N	4	N	FN2
	M113	Armored Personnel Carrier	Y	0	Y	FN3
	Zil-131	Small Budget Truck	N	4	N	FN4
	M35	Large Budget Truck	N	4	N	FN5
Out of Library	SA-8 TZM	SA-8 Reload vehicle	N	6	N	OOL1
	BMP-1	tank w/small turret	Y	0	Y	OOL2
	BTR-70	8-wheeled transport	N	8	N	OOL3
	SA-13	turret SAMs	Y	0	N	OOL4
	SA-8 TEL	integrated radar exposed SAMS	N	6	N	OOL5

The DCS collection consists of two-dimensional X-band SAR imagery. Table 9 lists the 15 targets contained in the collection. Ten targets are in-library targets. The classifiers are trained using feature data from these targets. The in-library targets are grouped into two classes, hostile and friend/neutral. The SCUD is labeled "target-of-the-day" (TOD) and is the focus of the ATR system. The remaining four hostile targets are labeled "other hostile" (OH). The five friend/neutral target types are labeled FN.

Five target types (SA-8 TZM, BMP-1, BTR-70, SA-13, and SA-8 TEL) are grouped into the out-of-library class. The signatures of these target types are not used to train the classifiers and are labeled OOL.

The DCS radar data is collected using HH and VV polarizations. In the two-sensor experiment described above sensor 1 uses HH-polarized data and sensor 2 uses VV-polarized data.

Training and test data are segregated by depression angle from the airborne sensor to the ground vehicles at the time of collection. Flight passes at approximately 3000 and 4000 ft. correspond to sensor depression angles of 6 and 8 degrees respectively. Data from these flight passes constitute training data. Table 10 lists the flight passes used for training data. Each flight pass images the complete target set across 90 degrees of aspect angle. Multiple flight passes provide imagery across 360 degrees of target aspect angle.

Test data is collected at a depression angle of 10 degrees to form an extended operating condition (EOC) relative to the training data. Flight passes corresponding to 10 degrees of depression angle are made at approximately 5000 ft. of elevation. Table 11 lists the flight passes used to form the test set.

5.2.1 Features

Once grouped into sets according to sensor (polarization, either HH or VV), and training/test (6 and 8° depression angle for training and 10° depression angle for test), the SAR chips are processed into HRR profiles per the steps outlined in Section 2.2.6. These steps include:

- remove of Taylor windowing and oversampling in the DCS SAR chip
- apply inverse 2-D FFT
- convert to range domain
- form a mean HRR profile

Each 256×256 pixel SAR chip is processed into a 322-bin mean HRR profile. Features are extracted from each profile according to a maximum-value-within-bin-window rule. Each HRR profile is divided into 10 bin windows near the center of the profile as shown in Fig. 49. These windows are defined by HRR bin ranges as follows: 103-114, 115-126, 127-138, 139-150, 151-162, 163-174, 175-186, 187-198,

Table 10. Data Selected for Training with a Desired Depression Angle of 6 or 8 Degrees

Number	Flight	Pass	Identifier	Chips	Looks per vehicle	Desired dep angle
1	1	10	FP0110	690	46	6
2	1	11	FP0111	660	44	6
3	1	12	FP0112	660	44	6
4	1	13	FP0113	660	44	6
5	1	15	FP0115	690	46	8
6	1	16	FP0116	690	46	8
7	1	17	FP0117	690	46	8
8	1	18	FP0118	690	46	8
9	1	34	FP0134	690	46	8
10	2	12	FP0212	660	44	6
11	2	13	FP0213	660	44	6
12	2	14	FP0214	690	46	6
13	2	16	FP0216	690	46	8
14	2	17	FP0217	690	46	8
15	2	18	FP0218	690	46	8
16	2	19	FP0219	690	46	8
17	2	32	FP0232	660	44	6
18	2	33	FP0233	660	44	6
19	2	34	FP0234	690	46	6
20	2	35	FP0235	660	44	6
21	2	36	FP0236	660	44	6
22	2	37	FP0237	660	44	6
23	2	38	FP0238	690	46	6
24	2	39	FP0239	660	44	6
25	3	6	FP0306	660	44	6
26	3	7	FP0307	690	46	6
27	3	8	FP0308	690	46	6
28	3	9	FP0309	690	46	6
29	3	11	FP0311	690	46	8
30	3	12	FP0312	690	46	8
31	3	13	FP0313	690	46	8
32	3	14	FP0314	690	46	8

num looks per vehicle	1448
HH looks per vehicle	724
VV looks per vehicle	724
Total number of chips processed	21720

Table 11. Data Selected for Test with a Desired Depression Angle of 10 Degrees

Number	**Flight**	**Pass**	**Identifier**	**Chips**	**Looks per vehicle**	**Desired dep angle**
1	1	20	FP0120	660	44	10
2	1	22	FP0122	660	44	10
3	1	23	FP0123	690	46	10
4	1	25	FP0125	690	46	10
5	2	21	FP0221	660	44	10
6	2	23	FP0223	660	44	10
7	2	24	FP0224	660	44	10
8	2	26	FP0226	660	44	10
9	3	16	FP0316	660	44	10
10	3	18	FP0318	660	44	10
11	3	19	FP0319	660	44	10
12	3	21	FP0321	660	44	10
13	3	28	FP0328	690	46	10
14	3	29	FP0329	660	44	10
15	3	31	FP0331	660	44	10
16	3	32	FP0332	660	44	10
17	3	33	FP0333	660	44	10
18	3	34	FP0334	690	46	10
19	3	35	FP0335	690	46	10
20	3	36	FP0336	690	46	10

num looks per vehicle	892
HH looks per vehicle	446
VV looks per vehicle	446
Total number of chips processed	13380

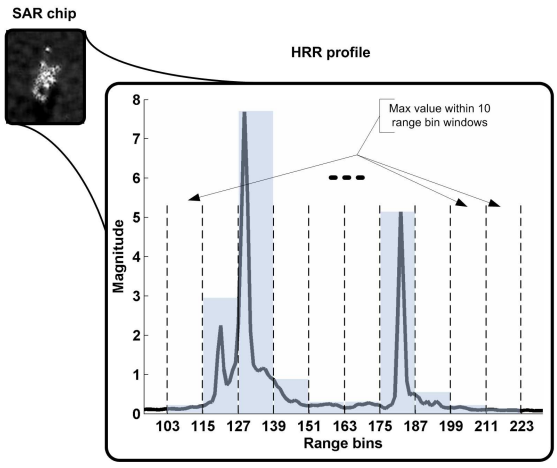

Figure 49. A SAR chip of a target at a specific sensor-target orientation is processed into an HRR profile. Features are derived from the profile by taking the maximum value within 10 range bin windows.

199-210, and 211-222. The maximum value within each of the 10 bin windows is saved as a feature. Thus dimensionality of each SAR chip is reduced from 256×256 to 10.

The HRR feature data is then ordered by aspect angle. Each 10-dimensional feature vector is derived from a target SAR chip and collected at a specific sensor-target orientation. This orientation includes both the depression angle from the airborne sensor to the ground vehicle and the relative aspect angle of the vehicle to the sensor line-of-sight in the horizontal plane. Variation in the depression angle separates the training and test data, and variation in the aspect angle determines the target pose. Observing a sequence of ordered target poses mimics a moving target or a stationary target and a moving sensor.

For a given target type the training data consists of 724 SAR chips processed into 724 HRR feature vectors. The 10-dimensional feature vectors are ordered by increasing aspect angle (from 1 to 360 degrees) and interpolated at 0.5 degree to form the complete training feature data. Figure 50 shows the feature data from

the training data set for the 10 in-library target types with hostiles on the left and friend/neutrals on the right.

Each subplot corresponds to a specific target type and displays the 720 interpolated 10-D HRR feature vectors in order of increasing target azimuth (aspect angle), where lighter colors correspond to greater magnitude and variation within a target as azimuth changes is apparent. Also, differences between target types given a 360 degree feature space representation are apparrent.

The time-series classifier used in this experiment acts on an ordered observation sequence of HRR feature vectors. The sequence begins at a random starting azimuth (aspect angle) and covers a subset of the 360 degree observations seen in Fig. 50. Figure 51 shows feature data for the 5 out-of-library target types.

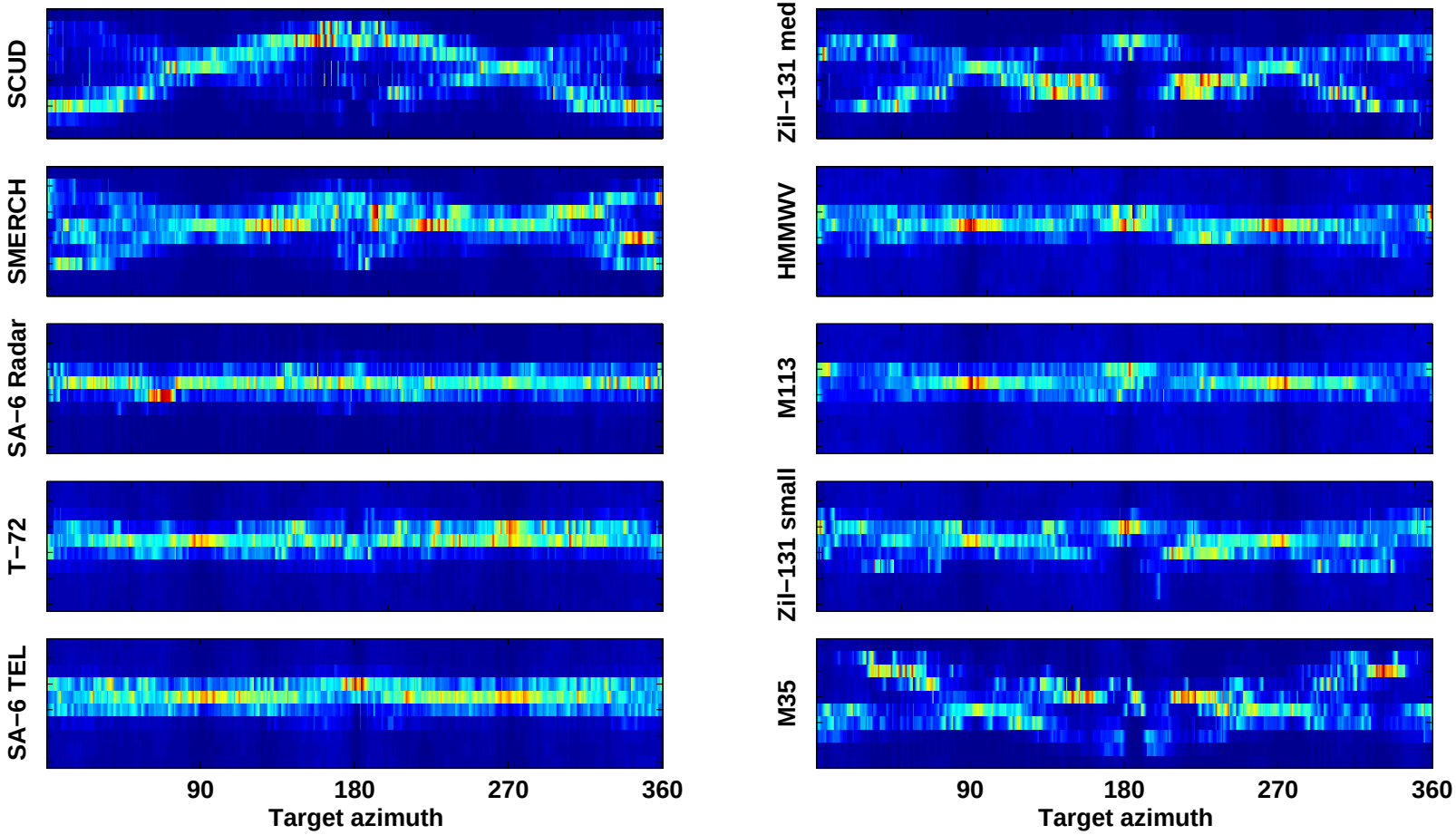

Figure 50. HRR-based feature training data for 10 in-library target types.

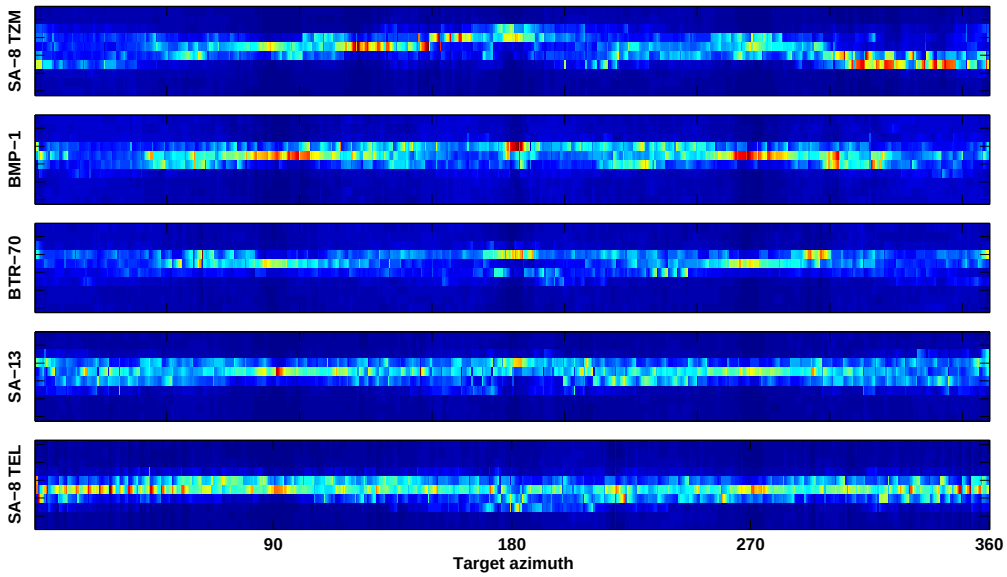

Figure 51. HRR-based feature test data for 5 out-of-library target types.

5.3 Classifiers

Section 5.2.1 describes how features are extracted from sensor data. Referring to Fig. 48, the sensor data D is processed into feature data F which is then input to a classifier c for labeling. This section describes the two types of classifiers used in the ATR system, a HMM-based classifier and a template-based classifier.

5.3.1 HMM-based classifier

The HMM-based classifier follows closely the development of the multi-dimensional Gaussian HMM of Section 3.4.4.3. For each target type, $t \in \{1, 2, 3, \ldots, 10\}$, an HMM λ_t is trained using sequences of 10-dimensional feature data. There are two sets of HMMs in the classifier. One set classifies feature data from sensor 1 (HH-polarized data) and is written λ_t^1, while another classifier set operates on the VV-polarized data of sensor 2, λ_t^2. Thus the HMM-based ATR system under investigation employs 20 HMMs operating on two streams of 10-dimensional time-series data.

A hidden Markov model λ is parameterized by the hidden Markov chain transition matrix A an observation distribution matrix B and an initial state probability vector π. Design of an HMM includes several decisions regarding its topology. Of critical importance is the number of hidden states in the Markov chain, called the order of the HMM. Given S states, the transition matrix is $S \times S$. Thus, the number of parameters in the HMM increases exponentially with S.

The HMMs used in this experiment are of order 90. The state space is not fully connected. To reduce the number of parameters and to more closely model the relationship between observations sequenced by target aspect angle and the observation distributions of each hidden state, the HMM uses a left-right state space (see Fig. 37). In a left-right model the Markov chain may remain in the same state or advance to the adjacent state (to the right) at each discrete time step. The state transition matrix A has entries on the main and first diagonal with zeros elsewhere, reducing the non-zero parameters of A from S^2 to $2S$.

Another topology decision is the modeling of the observation space. The observation space for this experiment is the 10-dimensional HRR feature vector derived from the HRR profile. A Gaussian HMM assumes the observation space is distributed multi-variate normal, where B_t^1 contains the parameter pair μ and $\mathbf{\Sigma}$ for the multi-variate Gaussian associated with each hidden state for the HMM associated with target t and sensor 1 and where μ is a 10-d mean vector and $\mathbf{\Sigma}$ is the 10-d covariance matrix. Since the HMM has 90 hidden states, B_t^1 is a 2×90 array, where the elements of the first column are the mean vector and covariance matrix associated with multi-variate Gaussian observation space of the first hidden state and where B_t^2 determines the observation distributions for the HMMs associated with sensor 2 data.

Prior to training using the Baum-Welch re-estimation algorithm, each model is given an initial parameterization such that

$$\lambda_t^s = (A_0^s, B_0^s, \pi),$$

where λ_t^s is the HMM for target t and sensor s, A_0^s is the initial state transition matrix, B_0^s is the initial observation distribution array, and π is the initial state distribution vector.

Initialization of the state transition matrix A_0^s makes use of the left-right model paradigm. The entries along the main and first diagonal are set to 0.5 and all other entries are set to 0, where A_0^s is a stochastic matrix with rows summing to 0.

The initial parameters for the observation distributions are also linked to the left-right model. As shown in Fig. 52 each hidden state observation distribution covers a window of target aspect angle. The elements of the initial observation distribution array B_0^s are found by determining the sample mean and covariance of the feature vectors within the aspect window of each hidden state. Because each λ_t^s has 90 states and the feature space includes observations from 1 to 360 degrees of aspect angle, each state covers an aspect window of 4 degrees.

Given F_t^s, the 10-dimensional feature data for target t and sensor s, the first column of entries of the observation distribution array B_0^s correspond to the mean vector and covariance matrix of the feature data from aspect angle 1 through 4 and are associated with the first hidden state

$$B_0^s(:,1) = \begin{bmatrix} \mu_{\mathbf{1}} \\ \mathbf{\Sigma_1} \end{bmatrix} = \begin{bmatrix} \text{mean}(F_t^s(:,1{:}4)) \\ \text{cov}(F_t^s(:,1{:}4)) \end{bmatrix},$$

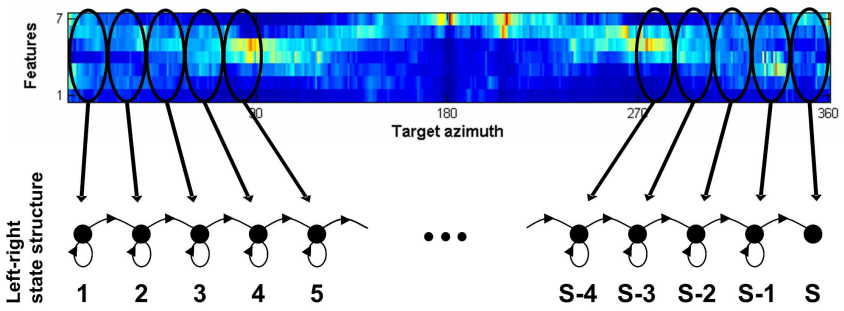

Figure 52. Multi-dimensional observations in the feature space are linked to the observation distributions of the hidden states. Note: features of dimension 7 are shown here; the DCS experiment uses features of dimension 10.

where the ":" notation indicates all elements of the specified dimension of the array. The last column of B_0^s is associated with the 90^{th} hidden state and is

$$B_0^s(:,90) = \begin{bmatrix} \mu_{\mathbf{90}} \\ \mathbf{\Sigma_{90}} \end{bmatrix} = \begin{bmatrix} \text{mean}(F_t^s(:,\, 357{:}360)) \\ \text{cov}(F_t^s(:,\, 357{:}360)) \end{bmatrix}.$$

The initial state distribution allows the user to control the starting state of the Markov chain. As described above, the observations are ordered by aspect angle beginning at 1 degree and ending with 360 degrees. The observations are a function of the aspect angle of the target; that is, when viewed from a certain aspect window, the observations come from a specific state in the hidden process. Given an observation sequence that begins at a target aspect angle of 1 degree, the model can be forced to start in state 1 by setting the first element of π to 1 with zeros elsewhere. Therefore, independent of target type t and sensor s the initial state distribution for the hidden state space always begins in state 1,

$$\pi_i = \begin{cases} 1 & \text{for} \quad i = 1 \\ 0 & \text{otherwise} \end{cases},$$

where i is the hidden state number.

With the HMMs initialized as described above, training ensues using the Baum-Welch re-estimation algorithm of Sec. 2.1.3.6, whereby the initial parameters of the HMMs are iteratively updated until a threshold is reached. Training records consist of the entire 10-dimensional feature data F_t^s for target t and sensor s. The training records begin with feature data at aspect angle 1, progress through the aspect window, and end at aspect angle 360. Once trained, each HMM (λ_t^s) is ready to be employed as a classifier in the experiment.

5.3.2 Template-based classifier

A competitor classifier using templates is described in this section. The classifier follows Laine [15], Meyer [60], and Duda et al. [20] in using Mahalanobis distance as a classification measure. Mahalanobis distance is

$$\Delta^2 = (\mu - \mathbf{x})^{\mathrm{T}} \mathbf{\Sigma^{-1}} (\mu - \mathbf{x}), \tag{65}$$

where μ is the population mean, $\mathbf{T}$ denotes matrix transpose, $\mathbf{\Sigma^{-1}}$ is the inverse covariance matrix of the population, and $\mathbf{x}$ is the test vector whose distance (squared) from the population is Δ^2.

Templates are formed using the 10-dimensional feature data for each target t and each sensor s. The feature data is divided into 24 wedges of 15 degrees, each covering the entire 360 degree aspect of the feature data. A sample mean and covariance are taken from each of the 24 wedges forming a template array T_t^s for each target type t and each sensor s. Descriptive statistics for the wedges are used to define the populations in the calculation of Mahalanobis distance.

The elements of the first column of the template array are determined by finding the mean and covariance of the feature vectors within the first aspect wedge

(1 through 15 degrees of aspect angle)

$$T_t^s(:,1) = \begin{bmatrix} \mu_{\mathbf{1}} \\ \mathbf{\Sigma_1} \end{bmatrix} = \begin{bmatrix} \text{mean}(F_t^s(:,1{:}15)) \\ \text{cov}(F_t^s(:,1{:}15)) \end{bmatrix}.$$

With the template arrays formed, each T_t^s is ready to be employed as a classifier in the experiment.

5.4 Methodology

At the heart of the extended CID optimization framework is the ATR system which labels observation sequences of unknown target type with one of five labels: target-of-the-day (TOD), other hostile (OH), friend/neutral (FN), out-of-library (OOL), or non-declare (Non-dec). Table 9 lists the 15 target types used in the experiment; 10 in-library types and 5 out-of-library types. This section describes the process of classification given trained HMM and template classifiers.

An HMM-based classifier consists of 20 models λ_t^s, where $t = 1, 2, \ldots, 10$ (in-library target types) and $s = 1, 2$ (sensors). Given a target under test, each sensor produces an observation sequence through a specific wedge of aspect angle. The observation sequences are processed into sequences of features. Feature sequences from sensor 1 are evaluated by the sensor 1 HMMs, λ_t^1, and sensor 2 sequences are evaluated by sensor 2 HMMs, λ_t^2. Classifier outputs are post-processed according to fusion rule and out-of-library discriminator before label thresholding occurs.

A similar process unfolds for the template-based classifier. The parameterized template arrays T_t^s are used to find a minimum Mahalanobis distance across target type when given test data. Classifier outputs are post-processed according to a fusion rule, an out-of-library assessment is made, and finally a label is assigned as a function of ROC and rejection thresholding.

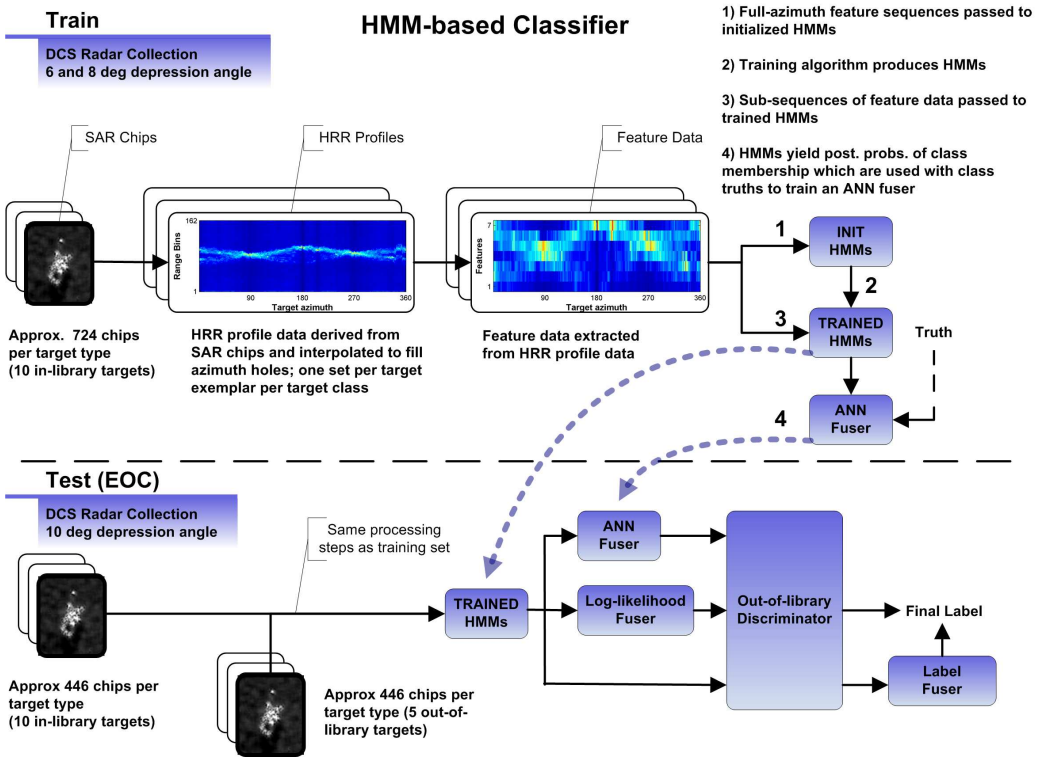

Figure 53. Experimental flowchart with HMM-based classifiers.

Figure 53 provides an experiment flowchart for the HMM case. The template case is similar with the exception of the classifier training routine. The following sections provide details of the classification methodology for the HMM-based and template-based classifiers.

5.4.1 Test sequence generation

As described in Sec. 5.2, the DCS data are segregated into training and test data as a function of depression angle. The data used to train the classifiers are collected at a depression angle of 6 and 8 degrees, while the test data are collected at 10 degrees. Test sequences are drawn from the ordered 10-dimensional feature data resulting from processing SAR chips into HRR profiles and then applying the maximum-value-within-bin-windows rule of Sec. 5.2.1. The same notation is used to

denote testing feature data as is used for training data, i.e., F_t^s, but $t = 1, 2, \ldots, 15$ indexes both in-library and out-of-library target types and s indexes sensors with the understanding that while the notation is the same, the feature data is not.

One hundred test records are generated for each in-library target type (10) and twenty test records are generated for each out-of-library type (5) for a total of 1100 test records. Let Y_t^{S} be an array containing the test sequences from target $t = 1, 2, \ldots, 15$ and sensor s. Because the interest is in time-series classification, each test record is an ordered sequence of feature observations. Each test record begins at a randomly chosen aspect angle and includes a pre-determined number of observations. For example, if the observation length is 10 degrees then each test record $y \in Y_t^{\mathrm{S}}$ begins at a randomly selected starting aspect angle and covers an aspect window of 10 degrees. Thus each y is a subset of the full aspect feature data for target t and sensor s. Each Y_t^{S} contains test sequence data and is presented to both HMM- and template-based classifiers for classification.

5.4.2 Classifier testing

Test records Y_t^{S} are presented to the HMM-based and template-based classifiers differently. The HMM-based classifiers λ_t^s are given the sequenced test data contained in Y_t^{S}. The methodology used in this experiment presents all 1100 sensor 1 test records (Y_t^1 for $t = 1, 2, \ldots, 15$) to each sensor 1 HMM (λ_t^1 for $t = 1, 2, \ldots, 10$). Each record is evaluated by each HMM for a total of 11,000 evaluations. The process is repeated for sensor 2 data.

Each test record is evaluated by 10 target-specific HMMs, producing 10 log-likelihoods per the calculations of Sec. 2.1.3.4. Class membership can be assigned at this point by choosing the model associated with the greatest log-likelihood among the 10. In this experiment, assignment of class membership is delayed until classifier output from both sensors is fused.

The template-based classifier is not a time-series classifier in the same sense as the HMM. Instead of taking a sequence of observation data as input, the template-based classifier takes a vector as input to the Mahalanobis distance calculation. Given a test record $y \in Y_t^{\mathrm{S}}$ that covers an aspect window of 10 degrees, the test vector $\mathbf{x}$ is formed by finding the mean of y. As in the HMM case, there are 1100 test records (vectors) in the template case.

Each test vector is used to find the Mahalanobis distance from each 15 degree aspect wedge of each target:

$$\Delta^2 = (\mu - \mathbf{x})^{\mathrm{T}} \mathbf{\Sigma^{-1}} (\mu - \mathbf{x}), \tag{66}$$

where T_t^s contains the μ and $\mathbf{\Sigma}$ for each aspect wedge of target t and sensor s.

The smallest Mahalanobis distance $\Delta_t^{\min}$ across the 24 aspect wedges for each target t are collected for each test record. Class membership can be assigned at this point by choosing the template associated with the smallest Mahalanobis distance among the 10. In this experiment, assignment of class membership is delayed until classifier output from both sensors is fused.

5.4.3 Classifier post-processing

Post-processing covers the steps from classifier output to classifier labeling. Included in these steps are

- Conversion from classifier output to estimated posterior probabilities given a test record
- Discrimination between in-library and out-of-library records given an estimated posterior probability of membership in the 10 in-library target types
- Labeling of class as a function of the ROC and rejection thresholds

The post-processing steps are described for the simple case of a single sensor without fusion. The HMM-based classifier system uses 10 models $\lambda_t^1 \quad t = 1, 2, \ldots, 10$, trained with feature data derived from sensor 1 (HH polarized SAR chips). Given a test record y, the 10 models output 10 log-likelihoods. The log-likelihoods are exponentiated and normalized to produce an estimated posterior probability for each target type pp_t for $t = 1, 2, \ldots, 10$.

In the template-based classifier case, the min-distance results $\Delta_t^{\min}$ are mapped to a [0, 1] interval using

$$z_t = \frac{1}{\sqrt{2\pi}}\, e^{-1/2\Delta_t^2} \tag{67}$$

and are then normalized across the 10 in-library target types into posterior estimates

$$pp_t = \frac{z_t}{\sum_{i=1}^{10} z_i} \qquad \text{for} \quad t = 1, 2, \ldots, 10. \tag{68}$$

Whether derived from HMM or template outputs, the posterior estimates pp_t are processed by an in-library/out-of-library discriminator in the same fashion. The purpose of the discriminator is to assign an 11^{th} posterior as a function of the 10 posteriors contained in pp_t. The 11^{th} posterior determines the estimated probability of membership in the out-of-library class.

Given the 10-D in-class posterior probability vector

$$\mathbf{x}_{\text{post}} = [pp_{\text{TOD}} \quad pp_{\text{OH1}} \quad pp_{\text{OH2}} \quad \cdots \quad pp_{\text{FN5}}],$$

the discrimination function sorts the posteriors in descending order, producing $\mathbf{x}_{\text{ord}}$. Assuming the classifier identifies in-library targets well, a small subset of the class posteriors is significantly larger than the remaining posteriors. In-library/out-of-library discrimination results from a threshold setting based on the sum of a subset of ordered posteriors.

Two threshold parameters are chosen through a nearly blind sub-optimization routine. The parameters are the number of ordered (largest to smallest) posteriors over which to sum, $\theta^{(1)}_{\text{OOL}}$, and the threshold $\theta^{(2)}_{\text{OOL}}$. The parameter values are chosen to ensure some minimum discrimination performance given in-library and out-of-library records, hence the nearly blind description. For example, the sub-optimization routine may determine a threshold $\theta^{(2)}_{\text{OOL}}$ based on the sum of the second through sixth ordered posteriors, $\theta^{(1)}_{\text{OOL}} = 6$. Thus given $\mathbf{x}_{\text{ord}}$ for a sample test record, the discriminator compares

$$\mathbf{x}_{\text{ool}} = \sum_{i=2}^{\theta^{(1)}_{\text{OOL}}} \mathbf{x}_{\text{ord}}(i),$$

the sum of the second through sixth ordered posteriors for the test record, with the threshold $\theta^{(2)}_{\text{OOL}}$ and assigns the **OOL** posterior as a function of the distance from the threshold

$$pp_{\text{OOL}} = \begin{cases} 0 & \text{if } \mathbf{x}_{\text{ool}} < \theta^{(2)}_{\text{OOL}} \\ f(\mathbf{x}_{\text{ool}} - \theta^{(2)}_{\text{OOL}}) & \text{if } \mathbf{x}_{\text{ool}} \geq \theta^{(2)}_{\text{OOL}} \end{cases}.$$

If $\mathbf{x}_{\text{ool}} < \theta^{(2)}_{\text{OOL}}$, then the record is from an in-library class and the posterior probability for OOL is set to zero. If $\mathbf{x}_{\text{ool}} \geq \theta^{(2)}_{\text{OOL}}$, then the record is potentially an out-of-library record and the posterior probability for OOL is set to a monotonically-increasing function of the distance from the threshold:

$$f(d) = \frac{2}{1 + e^{-10d}},$$

where $d = \mathbf{x}_{\text{ool}} - \theta^{(2)}_{\text{OOL}}$. Since $\mathbf{x}_{\text{ool}} \in [0, 9/10]$ and $\theta^{(2)}_{\text{OOL}} \in [0, 1]$, then $d \in [0, 9/10]$ and f maps d to $[1, 1.999]$. Finally pp_{OOL} is concatenated to the end of the 10-element estimated posterior vector $\mathbf{x}_{\text{post}}$ and normalized to produce the estimated 11-element posterior probability vector.

The final step is assigning one of five labels to the test record. The five labels (TOD, OH, FN, OOL, or Non-declare) are assigned as a function of the 11-dimensional posterior probability vector $\mathbf{x}_{\text{post}}$ and the threshold settings θ_{ROC} and

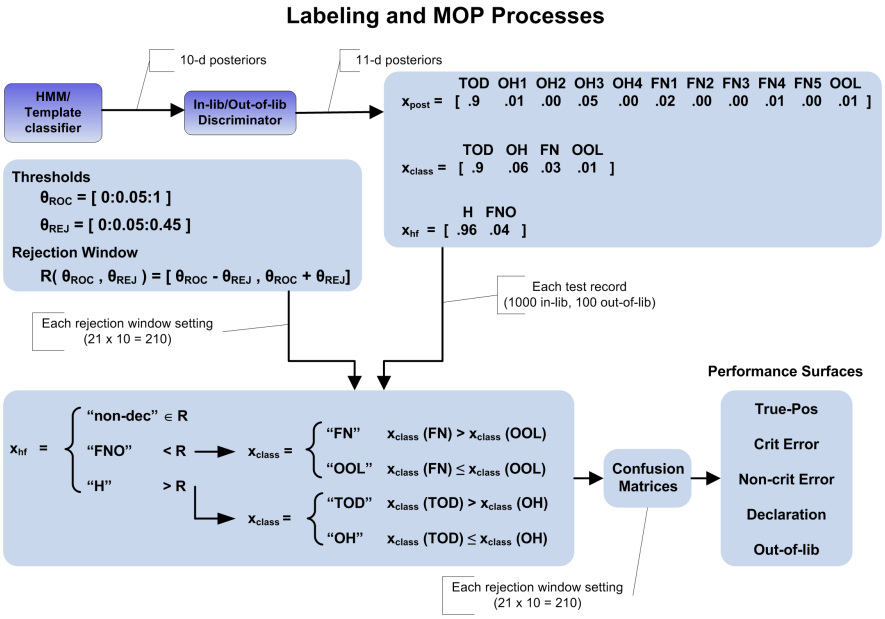

Figure 54. Labeling process and measures of performance (MOP) for the DCS experiment as a function of θ_{ROC} and θ_{REJ} thresholds.

θ_{REJ}, which define the rejection region. Figure 54 provides a roadmap for the labeling process.

As shown in the figure, the 11-dimensional posterior probability vector $\mathbf{x}_{\text{post}}$ is converted to a four-class $\mathbf{x}_{\text{class}}$ and a two-class $\mathbf{x}_{\text{hf}}$ posterior vector by summing the posteriors related to the four true target classes (TOD, OH, FN, and OOL), and finally separating the posteriors into two super-classes (H = TOD + OH, and FNO = FN + OOL).

A rejection region is determined by θ_{ROC} and θ_{REJ}. The two-class posterior vector $\mathbf{x}_{\text{hf}}$ is adjudicated with the rejection region, resulting in either a hostile declaration, a friend/neutral/out-of-library declaration, or a "Non-declare" label. If a hostile declaration is made, the associated four-class posterior vector $\mathbf{x}_{\text{class}}$ is adjudicated to determine whether the test record is assigned a "TOD" or "OH" label. If a friend/neutral/out-of-library declaration is made, the associated four-class posterior

vector $\mathbf{x}_{\text{class}}$ is adjudicated to determine whether the test record is assigned a "FN" or "OOL" label.

Each test record is evaluated at a specific (θ_{ROC}, θ_{REJ}) setting. A confusion matrix is built using label versus truth for the test records at each threshold setting. Performance measures are collected per the calculations of Sec. 4.3.2.

5.4.4 Fusion methods

The DCS experiment makes use of three different fusion methods to combine the classifier outputs of the two sensors. The first two fusion methods, mean fusion and neural network fusion, combine the classifier outputs prior to labeling. Rejection region thresholding is applied to the fused 11-dimensional posterior vector $\mathbf{x}_{\text{post}}$. In the third case, label fusion, each classifier output is adjudicated by the rejection region producing one of five labels (TOD, OH, FN, OOL, or Non-declare). Two sets of labels are produced, one by classifier 1 and another by classifier 2. The labels are fused according to a set of label rules that map all possible label pairs into a final fused label. This section examines the three methods of fusion.

Figure 55 shows the process for the fusion methods with the simple mean fusion rule on top. In the HMM case, given a test record, each set of sensor-specific HMMs λ_t^s produces a 10-dimensional vector of log-likelihoods. The mean fusion rule simply finds the mean of the two 10-dimensional log-likelihood vectors. The 10-dimensional mean vector is then exponentiated and normalized to produce a 10-dimensional estimated posterior probability vector. After adding an 11^{th} posterior via the in-library/out-of-library discriminator, the posterior vector $\mathbf{x}_{\text{class}}$ is adjudicated according to the rejection region thresholds, producing a final label.

In the case of template classifiers, the mean fusion rule is applied to the 10-dimensional minimum Mahalanobis distance vectors $\Delta_t^{\min}$ associated with each sensor. Here the mean of the two min-distance vectors is produced by the mean fusion

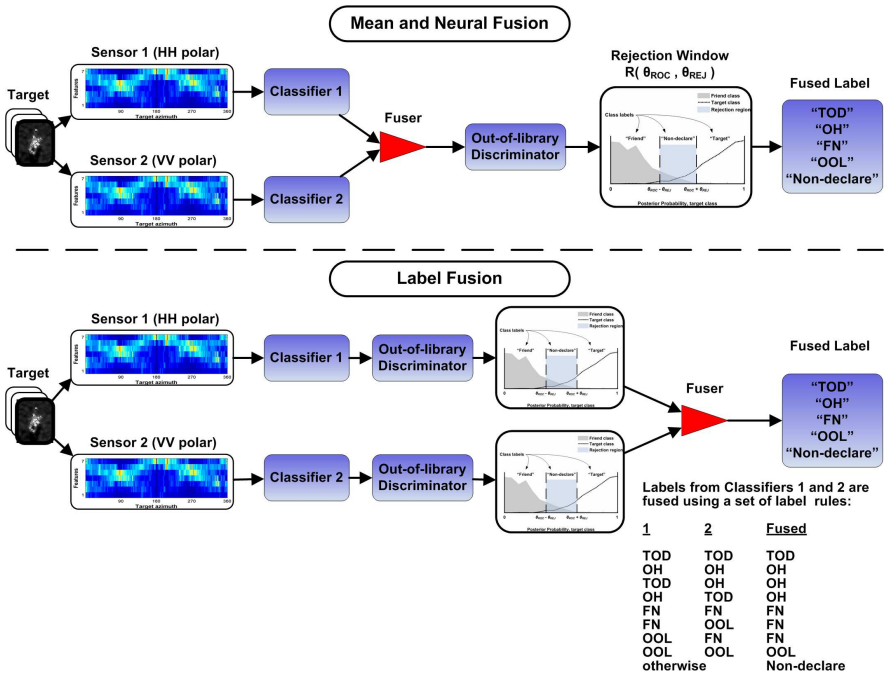

Figure 55. Fusion methods

rule. The mean vector is then mapped to the interval [0, 1] and normalized into a 10-dimensional estimated posterior probability vector. After adding an 11^{th} posterior via the in-library/out-of-library discriminator, the posterior vector is adjudicated according to the rejection region thresholds, producing a final label.

The neural network fusion method is similar to the mean fusion method. It takes 10-dimensional classifier output from the two sensors and produces a single fused 10-dimensional vector. Instead of a simple mean rule, the neural network fusion rule employs a multi-layer perceptron neural network (MLPNN) to fuse the two sets of inputs.

The MLPNN takes an input vector comprised of the two 10-dimensional classifier output vectors (either log-likelihood in the case of HMMs or min-distance for the template case) concatenated to form a vector of length 20. The trained MLPNN then maps the input vector to a 10-dimensional output vector whose entries are in the range [0, 1].

The structure of the MLPNN used in the experiment has 20 input nodes, 40 hidden layer nodes, and 10 output nodes. A tansigmoid transfer function is used for the hidden layer while a logsigmoid transfer function is used at the output layer. The input data is pre-processed to the range $[-1, 1]$.

Training of the MLPNN uses sequences from the training data set (6 and 8 degree depression angle) to produce output from the HMM and template-based classifiers. These outputs are used as training input for the MLPNNs. The inputs are targeted against the known true-class of the input vectors. MLPNN training uses a gradient-descent method with momentum to determine network weights and biases.

The final method is label fusion. As mentioned earlier, the label method combines labels instead of classifier outputs. Figure 55 shows the label fusion process and includes the set of label rules used in the experiment.

The threshold space used in the label fusion rule is quadratically larger than the other fusion rules. This result follows from performing rejection region adjudication for each classifier, which squares the number of threshold settings.

5.4.5 Prior knowledge of target aspect

One factor influencing classifier performance is prior knowledge of the target aspect angle. In the case of the template-based classifier, prior knowledge of target aspect angle reduces the number of aspect wedges involved in the minimum Mahalanobis distance calculation. If target aspect is known so that the target aspect angle falls within a specific aspect wedge, then the min-distance calculation is reduced from 24 wedges per target to 1 wedge per target, decreasing the chance for classifier error.

In Laine's research [15], prior aspect knowledge is assumed to be within $\pm$ 22.5° due to target tracking information handoff to the ATR. Using this level of prior target

Table 12. Prior aspect distribution for HMM ATR

Number HMM States	**Aspect Wedge**	**Half-Width**	**States to cover** $\pm 22.5^\circ$
90	4° per state	$\pm 2^\circ$	11 states ($\pm 22^\circ$)
72	5°	$\pm 2.5^\circ$	9 states ($\pm 22.5^\circ$)
60	6°	$\pm 3^\circ$	8 states ($\pm 24^\circ$)
40	9°	$\pm 4.5^\circ$	5 states ($\pm 22.5^\circ$)
30	12°	$\pm 6^\circ$	4 states ($\pm 24^\circ$)
20	18°	$\pm 9^\circ$	3 states ($\pm 27^\circ$)
10	36°	$\pm 18^\circ$	1 state ($\pm 18^\circ$)

aspect information corresponds to 3 aspect wedges ($3 \times 15^\circ = 45^\circ$) in the case of the template-based classifier. Thus, when searching for the min-distance Mahalanobis measurement, only the true wedge and its nearest neighbor on either side are used.

For the HMM-based classifier, ensuring a specific level of target aspect angle knowledge is more problematic. The solution makes use of the relationship between hidden states and aspect angle windows. Table 12 lists the number of states associated with a $\pm$ 22.5° aspect window given that there are S states in the Markov chain. As S increases, the number of states required to cover the aspect window increases. For the case $S = 90$, 11 states are required to cover the aspect window.

Given a test sequence that begins at angle α and an HMM with 90 hidden states such that each hidden state is associated with an aspect window of 4°, α corresponds to a specific aspect window and hence a specific hidden state called s^*. With perfect prior knowledge of target aspect angle α, the HMM prior state distribution π used in the evaluation of the test sequence sets $\pi(s^*) = 1$ and 0 elsewhere. With imperfect knowledge, a uniform distribution is centered on $\pi(s^*)$. For $S = 90$ and aspect knowledge limited to $\pm$ 22.5° the uniform distribution covers 11 states centered on $\pi(s^*)$.

Analysis of the raw SAR image data reveals the $\pm$ 22.5° assumption to be achievable through simple image analysis. Figure 56 shows the steps used in this image processing analysis. The raw data is the collection of SAR chips of a specific target type (T-72 tank), each collected at a specific sensor-target orientation. The

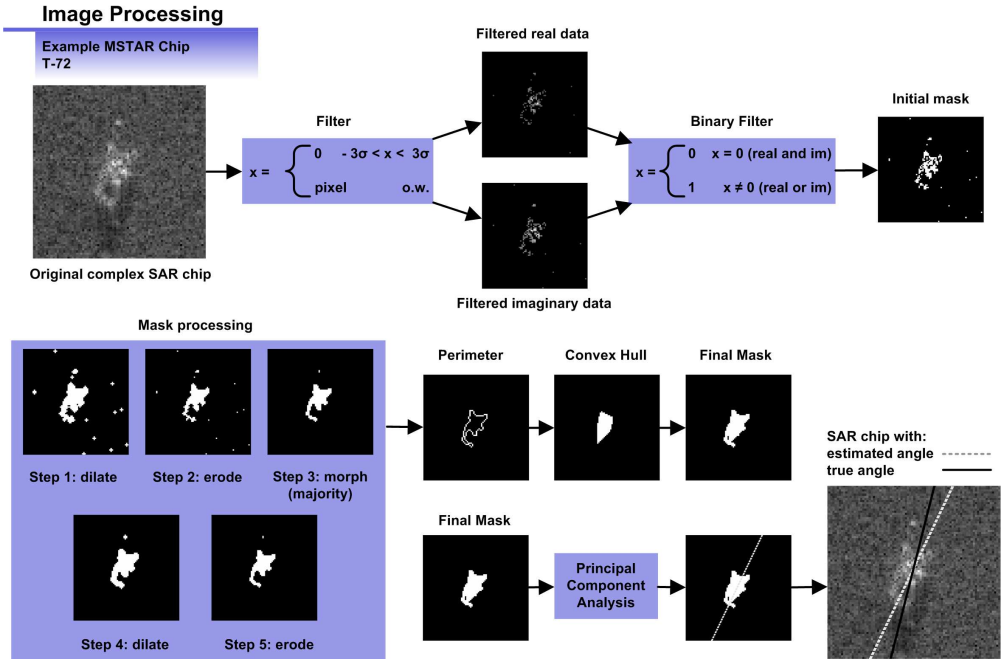

Figure 56. SAR chip image processing flowchart

SAR chips are ordered by target aspect angle, and each chip is processed according to the following steps:

- The complex SAR chip is separated into its component real and imaginary parts.
- Each sub-chip is filtered according to descriptive statics of the pixel values.
- A binary mask is created according to a boolean rule involving the filtered real and imaginary sub-chips.
- The initial mask is manipulated using various image processing routines.
- A final binary mask is created.
- Principal component analysis (PCA) is performed on the binary mask to yield a major axis of the mask which is used to estimate the target aspect angle.
- The true aspect angle is compared with the PCA-derived estimate.

The application of PCA to the pose-estimation problem begins with the binary target mask B based on the 128×128 pixel complex SAR chip. Thus B is a 128×128 matrix with elements $b_{ij} \in \{0, 1\}$. Next, the row and column locations of the elements of B where $b_{ij} = 1$ are placed in C, a $n \times 2$ matrix, where n is the number of target pixels in the target mask.

Principal component analysis is then performed on the two-dimensional data of C. First, the column-wise means are subtracted from the data, leaving the centered data C_0. Next, the normalized covariance matrix $\mathbf{\Sigma}$ of C_0 is determined. Then the eigenvectors $\mathbf{x}$ of $\mathbf{\Sigma}$ are found as solutions of

$$(\mathbf{\Sigma} - \lambda \mathbf{I})\mathbf{x} = 0,$$

where $\mathbf{I}$ is the identity matrix and λ are the eigenvalues associated with the eigenvectors $\mathbf{x}$. Since C_0 is two-dimensional, the eigenvector associated with the largest eigenvalue is the major axis of target mask pixels. Intersecting the major axis with the target mask centroid yields the estimated target aspect angle. Figure 57 shows a sequence of six SAR chips with the PCA-estimated and true target aspect angles.

The processing steps are repeated for each available chip. Figure 58 shows the distribution of errors from the aspect angle estimation experiment. A mean error of 11.34 is within the assumed $\pm$ 22.5° accuracy of the previous discussion.

Pose estimation in the context of ATR is not new [101, 102, 103, 104, 105]. Two approaches exist in research relative to pose estimation. The first employs adaptive classifiers that must be trained before implemention. An example of this type of pose estimator is a neural network [101, 103]. The other uses image processing techniques to segment the target, then applies various criteria to estimate the target pose [102, 104, 105].

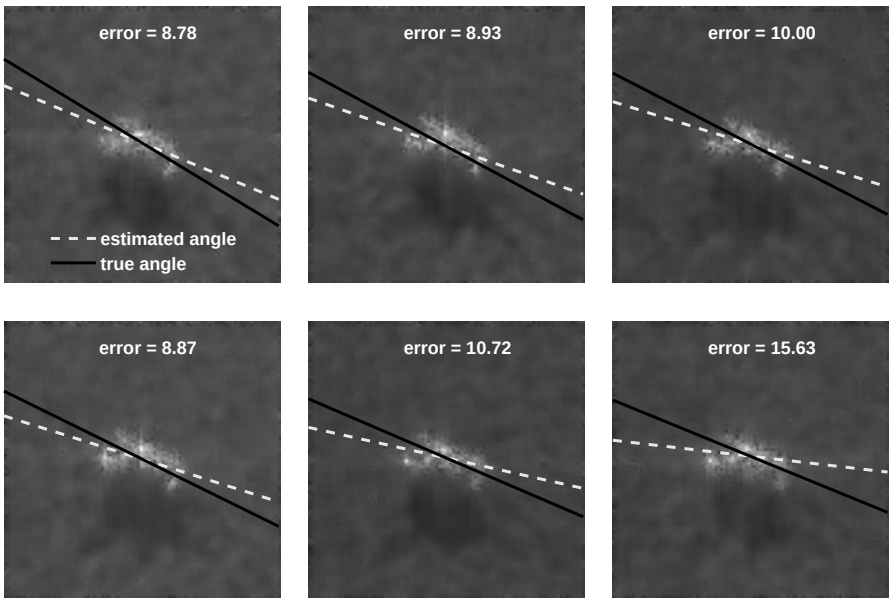

Figure 57. Sequence of six T-72 SAR chips from the MSTAR publicly-available collection with estimated and true angles indicated.

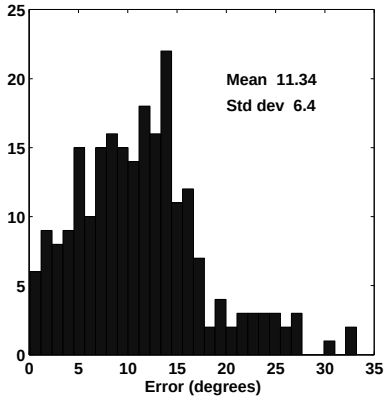

Figure 58. Distribution of errors when estimating target azimuth using principal component analysis on a target mask. Here 231 samples of the T-72 main battle tank from the MSTAR data collection (17 deg depression angle, target identification SN812) are used.

5.4.6 Target class prevalence

The DCS experiment uses 100 test records of each in-library target type. Since there are 5 hostile target types and 5 friend/neutral target types, the ratio of hostile to friend/neutral is 1:1. One hundred additional records are used to test the classifiers against out-of-library target types. The ratio of in-library to out-of-library target records is 10:1. These class prior probabilities impact classifier performance by simulating operation in a target-rich, target-sparse, or target-friendly equivalent environment.

The impact on system performance of varying target class prevalence is explored in the experiment of Sec. 5.6.2.

5.4.7 Correlation of observations

The DCS experiment assumes that both sensors are located on the same observation platform. Indeed, the DCS data are collected from the same sensor using two polarizations. For the purposes of the experiment, the data are presumed to have come from two different sensors located on the same platform. Thus, the starting aspect angle of an observation sequence from sensor 1 results in the same starting aspect angle for sensor 2. The observation data from sensor 1 and sensor 2 are correlated in that they observe the target from a shared orientation across the observation window.

The impact of altering sensor location on system performance is explored in the experiment of Sec. 5.6.2.

5.5 Extended CID optimization framework

This section presents the formulation for the extended CID optimization framework used in the DCS experiment and defines the pertinent performance measures.

5.5.1 Formulation

The formulation follows closely that presented in Sec. 4.3.3, where $\mathbf{x}$ is a vector of decision variables defined in the MP formulation.

Objective Function:

$$\max_{x \in X} \text{TPR}(x) = \frac{P_{\text{tp}}(x)}{\text{num looks}} \qquad \text{maximizes true-positive rate} \tag{69}$$

Subject to:

Warfighter constraints

$$\begin{aligned}
E_{\text{crit}} &< 0.1 && \text{upper bound on critical errors} \\
E_{\text{ncrit}} &< 0.2 && \text{upper bound on non-critical errors} \\
P_{\text{tp}} &> 0.85 && \text{lower bound on true-positive performance} \\
P_{\text{dec}} &> 0.5 && \text{lower bound on declaration performance} \\
P_{\text{ool}} &> 0.35 && \text{lower bound on out-of-library performance}
\end{aligned}$$

Fusion rule constraint

$$\sum_{i=1}^{f} F_i = 1 \qquad \text{select a single fusion rule}$$

$$\text{where} \quad F_i = \begin{cases} 1 & \text{if } i\text{th fusion rule used} \\ 0 & \text{otherwise} \end{cases}$$

Sensor selection constraint

$$\sum_{j=1}^{s} S_j \leq s \qquad \text{select from available sensors}$$

$$\sum_{j=1}^{s} S_j \geq 1 \qquad \text{select at least one sensor}$$

$$\text{where} \quad S_j = \begin{cases} 1 & \text{if } j\text{th sensor used} \\ 0 & \text{otherwise} \end{cases}$$

Threshold constraints

$$\theta^{ij} \geq 0 \qquad \text{lower threshold constraint}$$

$$\theta^{ij} \leq 1 \qquad \text{upper threshold constraint}$$

where θ^{ij} is the decision threshold associated with fusion rule i and sensor j. The decision threshold may be θ_{ROC} or θ_{REJ}.

5.5.2 Performance Measures

5.5.2.1 True-positive

The estimate for the true-positive performance measure is the number of true hostile records labeled "hostile" divided by the total number of true hostile records declared. This definition accounts for the added rejection option and the resultant "non-declare" label. The calculation is

$$\begin{aligned} P_{\text{tp}} &= P(\text{“TOD”} \cup \text{“OH”} | (\text{TOD} \cup \text{OH}) \cap \text{declaration}) \\ &= \frac{\text{num}(\text{“TOD”}|\text{TOD} + \text{“TOD”}|\text{OH} + \text{“OH”}|\text{TOD} + \text{“OH”}|\text{OH})}{\text{num}(\text{TOD declared} + \text{OH declared})}. \end{aligned} \tag{70}$$

5.5.2.2 Critical error

Critical error is

$$P(E_{\text{crit}}) = P\left(\left(\begin{array}{c} P(\text{“TOD”} \cap \text{FN}) \cup P(\text{“OH”} \cap \text{FN}) \cup \\ P(\text{“FN”} \cap \text{TOD}) \cup P(\text{“FN”} \cap \text{OH}) \end{array}\right) \mid \text{declaration}\right). \quad (71)$$

Simplification of Eq. 71 makes use of Bayes' rule, and depends on class prevalence as defined by class prior probabilities. Let $P(\text{TOD})$, $P(\text{OH})$, $P(\text{FN})$, and $P(\text{OOL})$ be the prior probabilities of the four true target classes, and $P(\text{“TOD”})$, $P(\text{“OH”})$, $P(\text{“FN”})$, $P(\text{“OOL”})$, $P(\text{“Non-declare”})$ be the unconditional system label probabilities, then

$$P(\text{TOD}) + P(\text{OH}) + P(\text{FN}) + P(\text{OOL}) = 1 \quad (72)$$

$$P(\text{“TOD”}) + P(\text{“OH”}) + P(\text{“FN”}) + P(\text{“OOL”}) + P(\text{“Non-declare”}) = 1. \quad (73)$$

The simplified $P(E_{\text{crit}})$ calculation is

$$P(E_{\text{crit}}) = \frac{\left(\begin{array}{c} P(\text{“TOD”}|\text{FN})P(\text{FN}) + P(\text{“OH”}|\text{FN})P(\text{FN}) + \\ P(\text{“FN”}|\text{TOD})P(\text{TOD}) + P(\text{“FN”}|\text{OH})P(\text{OH}) \end{array}\right)}{1 - P(\text{“Non-declare”})}, \quad (74)$$

where $P(\text{“Non-declare”})$ is determined by the sum of the class-specific probability of non-declaration

$$\begin{aligned} P(\text{“Non-declare”}|\text{TOD})P(\text{TOD}) &+ P(\text{“Non-declare”}|\text{OH})P(\text{OH}) \\ &+ P(\text{“Non-declare”}|\text{FN})P(\text{FN}) \\ &+ P(\text{“Non-declare”}|\text{OOL})P(\text{OOL}). \end{aligned} \quad (75)$$

5.5.2.3 Non-critical error

As with critical error, non-critical error calculation reverses the order of conditioning seen in the true-positive calculation. Non-critical error calculation involves vertical analysis of the confusion matrix.

Some flexibility exists in choosing which classification errors constitute non-critical errors. For this experiment, non-critical errors consider cross-labeled hostile targets (i.e., TOD labeled "OH" and OH labeled "TOD") and mis-labeling true classes as out-of-library:

$$P(E_{\text{ncrit}}) = P\left(\left(\begin{array}{c} P(\text{“TOD”}|\text{OH}) \cup P(\text{“OH”}|\text{TOD}) \cup \\ P(\text{“OOL”}|\text{TOD}) \cup P(\text{“OOL”}|\text{OH}) \cup \\ P(\text{“OOL”}|\text{FN}) \end{array}\right) | \text{ declaration}\right). \quad (76)$$

This result simplifies to the following non-critical error calculation incorporating prior class probabilities

$$P(E_{\text{ncrit}}) = \frac{\left(\begin{array}{c} P(\text{“TOD”}|\text{OH})P(\text{OH}) + P(\text{“OH”}|\text{TOD})P(\text{TOD}) + \\ P(\text{“OOL”}|\text{TOD})P(\text{TOD}) + P(\text{“OOL”}|\text{OH})P(\text{OH}) + \\ P(\text{“OOL”}|\text{FN})P(\text{FN}) \end{array}\right)}{1 - P(\text{“Non-declare”})}, \quad (77)$$

where $P(\text{“Non-declare”})$ is determined by the sum of the class-specific probability of non-declaration.

5.5.2.4 Declaration

The declaration performance measure captures the percentage of test records which the system labels with one of the true class labels. The complementary measure is the non-declaration performance measure. It tabulates the number of records labeled "Non-declare" by the system:

$$
\begin{aligned}
P_{\text{dec}} &= 1 - P(\text{“Non-declare”}) \\
&= 1 - \left(\begin{array}{r} P(\text{“Non-declare”}|\text{TOD})P(\text{TOD}) + P(\text{“Non-declare”}|\text{OH})P(\text{OH}) \\ +P(\text{“Non-declare”}|\text{FN})P(\text{FN}) \\ +P(\text{“Non-declare”}|\text{OOL})P(\text{OOL}) \end{array} \right). \quad (78)
\end{aligned}
$$

5.5.2.5 Out-of-library

The out-of-library performance measure is a true-positive labeling of “OOL” given an OOL record using horizontal analysis of confusion matrix entries. The estimate for the out-of-library performance measure is the number of true OOL records labeled “OOL” divided by the total number of true OOL records declared:

$$
\begin{aligned}
P_{\text{ool}} &= P(\text{“OOL”}|\text{OOL}) \\
&= \frac{\text{num}(\text{“OOL”}|\text{OOL})}{\text{num}(\text{OOL declared})}. \quad (79)
\end{aligned}
$$

Vertical analysis of out-of-library performance (mis-labeling of true classes as out-of-library) is included in the non-critical error performance measure, but may also be defined as a second type of non-critical error per warfighter preference.

5.6 Results

Results are in two sections. The first provides initial results for a specific set of experimental parameters. The second explores results of a designed experiment where the experimental parameters are varied.

5.6.1 Initial results

The experiment settings for the initial results include the design of the HMM and template classifier described in Sec. 5.3, the data processing and methodology of Sec. 5.2 and Sec. 5.4, and the CID framework of Sec. 5.5.

The classifier design, data preparation, and experimental methodology, place the two competing classifiers on equal footing. Both classifiers use the same 10-dimensional feature data extracted and interpolated from HRR profiles of sequenced SAR target images to train on the 10-class problem. Test sequences for the two classifier types are drawn from the same SAR data set collected at a depression angle of 10 degrees (versus 6 and 8 degrees for the training set).

Test sequences contain the same number of observations and are considered taken from co-located sensors. There are an equal number of hostile target test records and friend/neutral test records. The ratio of hostile to friend/neutral to out-of-library test records is 5:5:1.

Post-processing classifier output is handled in an equivalent manner. Fusion rules are applied the same way, and the out-of-library discriminator functions are applied the same way for the HMM-based classifier as for the template-based classifier. Labeling the test records as a function of the ROC and rejection region thresholds is performed in the same way for both classifiers.

Both classifiers are evaluated within the same CID optimization framework. The objective function is the same; it maximizes true-positive performance as a function of number of sensor observations. The warfighter constraints are held constant for both classifiers. The minimum true-positive performance is 0.85, he maximum critical error rate is 0.1, the maximum non-critical error rate is 0.2, the minimum declaration rate is 0.5, and the minimum out-of-library performance rate is 0.35. The formulas for determining these performance measures, as shown in Sec. 5.5.2, are applied in the same manner to both types of classifiers.

Table 13. HMM- and template-based system performance comparison

Classifier	**Fusion**	**Percent feasible**						**Mean feasible value**						**Opt val**
		tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
								0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
HMM	Sensor 1	0.50	0.99	0.50	0.75	0.72	0.23	0.96	0.01	0.09	0.84	0.58	0.91	0.9723
	Sensor 2	0.50	0.99	0.32	0.75	0.75	0.07	0.94	0.04	0.04	0.84	0.65	0.91	0.9556
	Mean	0.50	0.99	0.50	0.75	0.75	0.25	0.96	0.02	0.09	0.84	0.64	0.91	0.9625
	ANN	0.42	0.79	0.37	0.73	0.70	0.06	0.99	0.03	0.05	0.78	0.62	0.98	1.0000
	Label	0.62	1.00	0.85	0.53	0.50	0.12	0.98	0.02	0.07	0.66	0.51	0.92	1.0000
Template	Sensor 1	0.36	0.92	0.40	0.75	0.60	-	0.98	0.05	0.07	0.83	0.48	-	-
	Sensor 2	0.36	0.87	0.40	0.75	0.65	0.01	0.98	0.05	0.06	0.82	0.53	0.87	0.8893
	Mean	0.38	0.95	0.40	0.75	0.68	0.06	0.98	0.03	0.06	0.83	0.56	0.91	0.9557
	ANN	0.37	0.51	0.35	0.68	0.67	-	0.98	0.06	0.06	0.81	0.58	-	-
	Label	0.40	0.98	0.69	0.43	0.20	-	0.99	0.03	0.06	0.67	0.38	-	-

The threshold space over which system performance is examined is the same for both types of classifiers. The ROC threshold θ_{ROC} varies from 0 to 1 in 0.05 increments, leading to 21 settings. The rejection region half-width threshold θ_{REJ} varies from 0 to 0.45 in 0.05 increments, leading to 10 settings.

Results for the initial system comparison are shown in Table 13. Performance results are shown for both types of classifiers, HMM-based on top and template-based on bottom. Given a type of classifier, the results are broken down by fusion methodology: first, no fusion methodology is chosen and sensor 1 and 2 operate as independent classifiers, second, a simple mean fusion rule, third, neural network fusion, and finally boolean fusion.

Performance is measured in two ways. First, a measure of classifier robustness is used. Percent feasible refers to the percentage of settings in the threshold space that result in feasible performance given a certain warfighter constraint. For example, referring to Table 13, of the $21 \times 10 = 210$ threshold settings for (θ_{ROC}, θ_{REJ}), 50% result in feasible performance for the true-positive warfighter constraint ($P_{\text{tp}} > 0.85$) in the case of Sensor 1 acting alone with an HMM-based classifier.

This measure of robustness is applied to each of the five warfighter constraints (true-positive, critical error, non-critical error, declaration, and out-of-library) in addition to the jointly feasible measure of robustness. In the joint case, the robustness measure captures the percentage of the threshold space that produces feasible points across all five constraints simultaneously.

The second measure of performance is the mean feasible value. The mean feasible value is the average value among feasible points for a given performance measure. Again, referring to Table 13, the mean feasible true-positive performance for Sensor 1 acting alone with an HMM-based classifier is 0.96. The boldface values located directly underneath the performance measure labels are the right-hand side of the warfighter performance constraints. The mean jointly-feasible performance value is mean true-positive value of the jointly-feasible points. The optimal value is the maximum true-positive value of the jointly-feasible points. If there are no feasible points a '-' is placed in the table at that location.

Figures 59 and 60 provide an additional method to analyze system performance. Figure 59 shows system performance for an HMM-based classifier system employing an artificial neural network (ANN) fusion rule. Figure 60 shows system performance for a template-based classifier system using a similarly-trained ANN fusion rule.

Each figure contains six subplots which detail system performance for the five warfighter performance measures plus a sixth subplot which shows joint performance across the five measures. The xy plane in each subplot indicate the ROC and reject threshold settings. The surface above represents system performance for a given warfighter performance measure, such as true-positive rate. Dots are used to indicate the threshold coordinate pairs where performance met the warfighter constraint (feasible points).

The sixth subplot shows the threshold coordinate pairs that are jointly-feasible across the five measures and plots them on the true-positive surface. The maximum

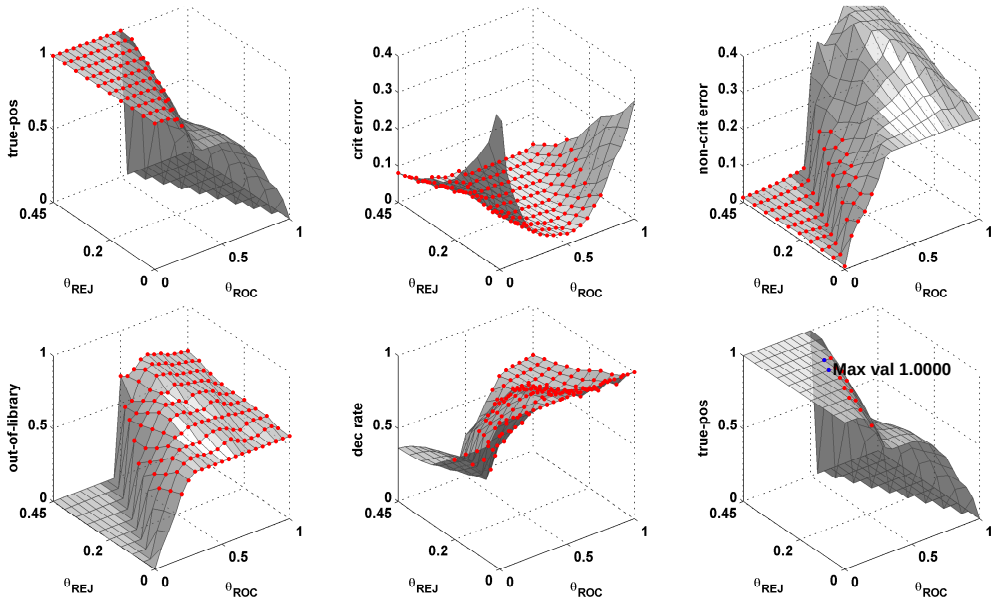

Figure 59. Performance surfaces determined by ROC threshold and reject threshold settings with feasible points shown for true-positive rate (top left), critical error rate (top middle), non-critical error rate (top right), out-of-library rate (bottom left), and declaration rate (bottom middle). Bottom right uses the true-positive surface to show jointly-feasible points with optimal point indicated. The system uses an HMM-based classifier, co-located sensors, and a neural network fusion method.

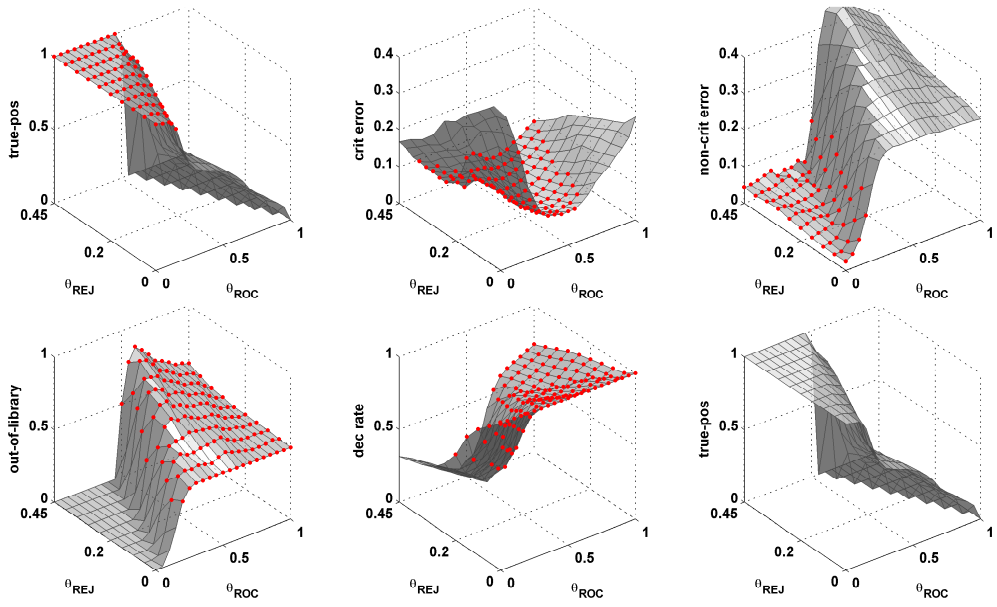

Figure 60. Performance surfaces for template-based classifier at same settings as HMM-based classifier of Fig. 59. Note the lack of jointly-feasible solutions in the bottom-right subplot.

value of the jointly-feasible points is also indicated. This value corresponds to the optimal value in Table 13.

5.6.1.1 Interpretation

Table 13 provides a concise collection of performance measures used to compare both classifier types and the methods used to fuse classifier outputs. To compare the classifier types, a mean value across fusion method is shown for both the robustness measures and the feasible values in Table 14. Clearly, the HMM-based classifier is more robust in the threshold space than the template-based classifier. Indeed, 15% of the threshold space yields jointly-feasible operating points for the HMM-based classifier, while this value is only 1% for the template-based classifier. When comparing the systems based on mean feasible performance measure values, the lack of jointly feasible operating points brings the template-based mean value down to

Table 14. Comparison of mean performance for HMM- and template-based systems

Classifier	**Percent feasible**						**Mean feasible value**						**Opt val**
	tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
							0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
HMM	0.51	0.95	0.51	0.70	0.68	0.15	0.96	0.02	0.07	0.79	0.60	0.93	0.9781
Template	0.37	0.85	0.45	0.67	0.56	0.01	0.98	0.04	0.06	0.79	0.50	0.36	0.3690

0.36, while the HMM classifier performs at 0.93. The mean optimal value also favors the HMM-based classifier at 0.9781 versus 0.3690.

When comparing fusion methods within a classifier type, some trends are observed. Referring to Table 13, the boolean fusion method provides better than average robustness for true-positive, critical error, and non-critical error, but its robustness lags for declarations and out-of-library feasibility. The mean performance values for the boolean case follow a similar trend, strong in true-positive, critical error, and non-critical error, but weak in declarations and out-of-library performance.

The performance surface plots shown in Figs. 59 and 60 capture results for the artificial neural network (ANN) fusion method for the HMM-based and template-based cases respectively. The plots reveal an HMM advantage in robustness in the critical error performance measure (middle subplot, top row). The HMM surface is lower (less critical error), resulting in more feasible points (79% versus 51% for the template-based classifier). The reduced feasible critical error space for the template-based classifier is the limiting factor in determining the lack of jointly-feasible operating points.

Two more plots similar to Figs. 59 and 60 are given to show performance surfaces for the case of the simple mean fusion rule. Figure 61 shows surface plots for the HMM-based case and the template-based case.

Comparing the plots of Fig. 61 with those of Figs. 59 and 60 yields several observations. First, a comparison of the surfaces for the HMM-based neural fusion

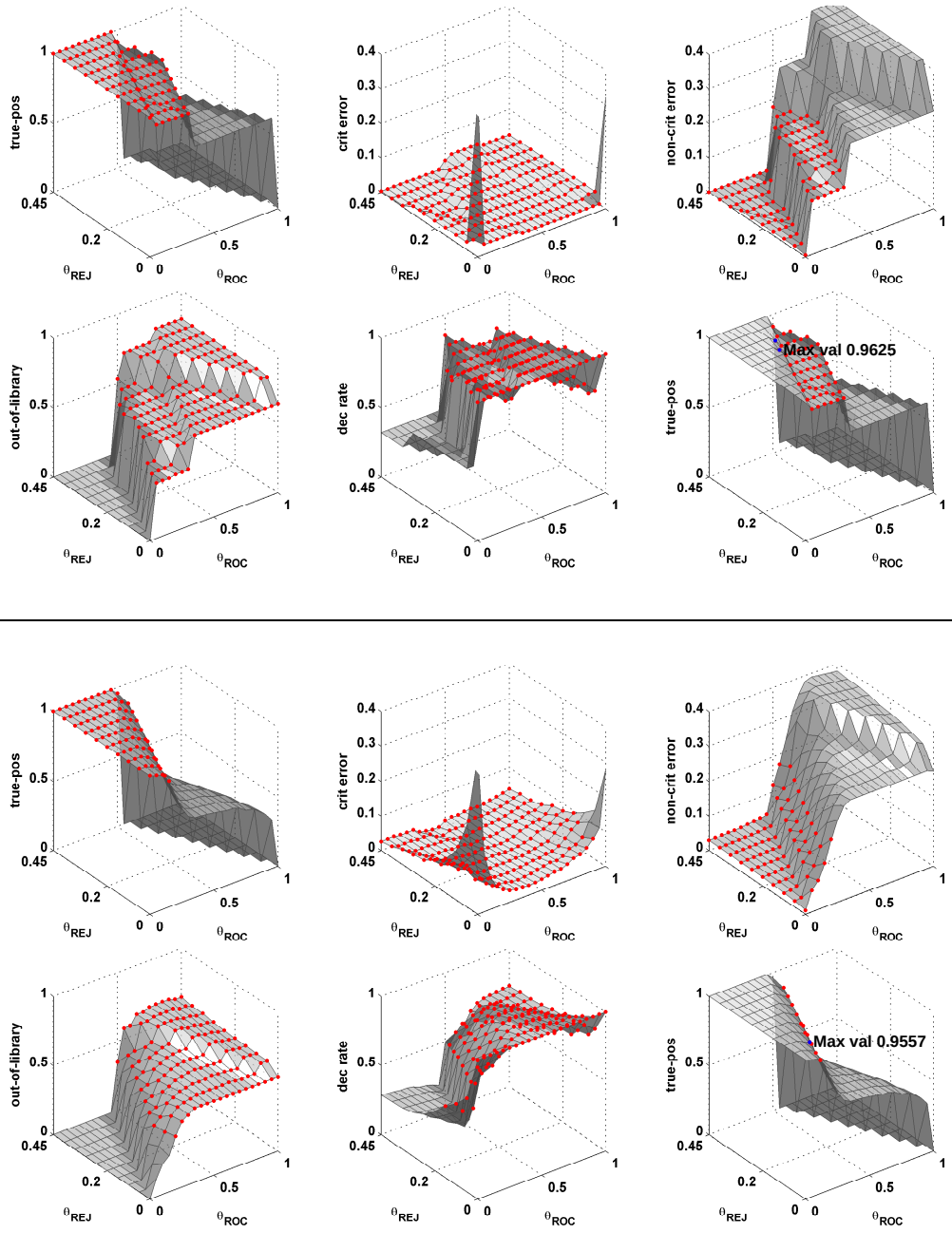

Figure 61. Performance surfaces determined by ROC threshold and reject threshold settings. HMM-based classifier surfaces are shown above the line, and template-based surfaces are shown below the line. Both experiments use co-located sensors and a simple mean fusion method.

and mean fusion shows more feasible space in the true-positive, critical error, non-critical error, and jointly-feasible surfaces for the mean fusion method. However, the neural fusion method has a perfect 1.0 optimal solution versus the mean fusion value of 0.9625 due to a more robust feasible space in the out-of-library performance surface. Further, the mean fusion method produces performance surfaces exhibiting sharp steps rather than the smooth contours of the neural network fusion. This result stems from the neural network mapping from classifier outputs to a [0,1] posterior space, producing a more evenly distributed posterior vector compared to the exponentiated log-likelihood of the mean fusion rule, which produces posteriors grouped tightly near either 0 or 1.

The template-based mean fusion performance surfaces shown below the line in Fig. 61 reveal an improved feasible space for the critical error measure. The more robust critical error feasible space allows several jointly-feasible solutions with an optimal value of 0.9557.

General trends evident no matter the classifier type or fusion method include trade-offs between the performance measures as a function of location within the threshold space. The best true-positive performance occurs in the northwest corner of the threshold space (looking down at the xy plane with (0,0) at southwest corner). This location corresponds to a low ROC threshold (aggressive hostile declaration) and a high rejection region threshold (large rejection region - only highly confident records are labeled).

The out-of-library performance surface rises where the true-positive surface falls, in the northeast corner of the threshold space. The best out-of-library performance occurs when the ROC threshold is high (conservative hostile declaration) and the rejection region is large. At this point very few hostile records are declared and the true-positive surface is at 0 in each of the plots.

Critical error peaks are at the southwest and southeast corners, where the rejection region nears 0 (few non-declarations) and the ROC threshold is near 0

(aggressive hostile declaration) or 1 (conservative hostile declaration). The saddle shape of the performance surface reveals whether the classifier-fusion pairing is robust in the critical error sense. If the saddle is low and flat (HMM-mean fusion), then the critical error performance is good. If the saddle is high with large sides (template-neural fusion), then the performance is poor.

Declaration performance is a function of the rejection region; the larger the rejection region, the greater the number of non-declarations, and hence the lower the declaration rate. The largest rejection region occurs when the ROC threshold is at 0.5 and the rejection region half-width threshold is at 0.45. This result yields a rejection region width of 0.9 centered at 0.5. Most plots reach a minimum declaration performance at this location in the threshold space. The HMM-mean fusion plot of Fig. 61, however, shows a relatively large declaration rate (approximately 0.8) at $(\theta_{\text{ROC}}, \theta_{\text{REJ}}) = (0.5, 0.45)$. This result is explained by the tight grouping of posteriors at 0 and 1 (outside the rejection window) resulting from the exponentiation of large negative log-likelihoods from the HMM classifiers.

Non-critical error incorporates cross-labeled hostile types ("TOD" for OH, "OH" for TOD) as well as in-library targets mis-labeled as out-of-library records. Thus, the non-critical error surface is influenced by the true-positive and out-of-library performance measures. In the northwest corner of the threshold space, true-positive performance is excellent and out-of-library performance is poor. Many hostile records are labeled correctly and few records are labeled as out-of-library. It is not surprising that the non-critical error surface is at or near 0 for this corner of the threshold space. As true-positive performance falls and more out-of-library labels are made, the non-critical error surface climbs rapidly.

Table 15. Designed experimental settings

Design parameter	**Settings**	**Purpose**
Classifier type	HMM-based template-based	competing classifiers
Fusion method	Sensor 1 Sensor 2 simple mean fusion ANN fusion label fusion	explore fusion methodologies
Sensor location	Co-located sensors Independent sensors	explore correlation of observations
Observation length	short medium long	explore effects of fewer/more observations
Target class prior probabilities (H:F)	10:1 4:1 2:1 1:1 1:2 1:4 1:10	explore target rich versus target sparse operating environments
Prior target aspect knowledge	$\pm 22.5^\circ$ $\pm 37.5^\circ$ none	explore effects of more accurate initial target aspect knowledge

5.6.2 Designed experiment results

The results shown and discussed in Sec. 5.6.1 correspond to specific experiment settings. These settings include the prior probabilities of the target classes, the location of the sensors, the number of observations in the test sequences, and the level of prior knowledge of target aspect angle. For the initial experiment the prior probabilities of the target classes are held at 1:1 for the hostile to friend/neutral classes. Sensors 1 and 2 are co-located, meaning they observe the target from the same orientation. The initial experiment assumes prior knowledge of the target aspect angle to within $\pm 22.5^\circ$.

This section describes a designed experiment by expanding the settings used in the initial experiment. Table 15 shows the designed experimental settings.

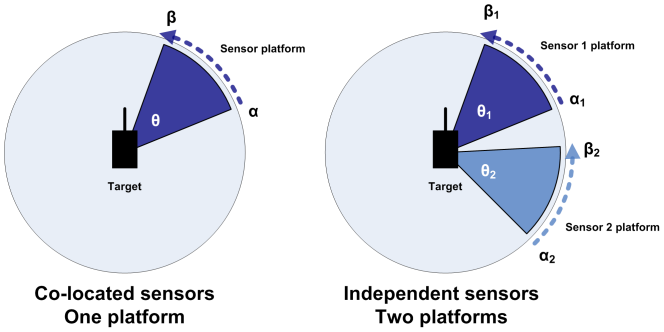

Figure 62. Co-located sensors on a single platform sweep out an observation window θ beginning at α and ending at $\beta = \alpha + \theta$. Independent sensors are located on separate platforms and sweep out observation windows $\theta_1 = \theta_2$ beginning at different starting angles $\alpha_1 \neq \alpha_2$.

As in the initial experiment, CID systems based on two types of classifier are in competition. The multiple classifier systems are fused using the same set of fusion methods as the initial experiment.

The designed experiment includes an additional sensor location setting. Originally, the sensors are co-located. The designed experiment allows the sensors to be located on different platforms, which means that the sensor-to-target orientation is no longer directly correlated. Rather, the observation sequences may begin a different target aspect angles for each sensor, which corresponds to two sensors simultaneously observing the same target from two different orientations as shown in Fig. 62.

The medium observation length is that used in the initial experiment. The designed experiment uses a reduced observation sequence (short) and an extended observation sequence (long) to explore the effect of more observations on classification performance. Intuitively, more observations means more data on which the classifiers can act, which should reduce error. However, the feature data is a noisy function of aspect angle, and a longer observation sequence may introduce more target class confusion compared to a short observation sequence.

The warfighter is interested in CID system performance across a variety of operational conditions. One of these conditions is the prevalence of hostile targets relative to friendly or neutral objects. The initial experiment held the prior probability of the hostile classes equal to the prior probability of the friend/neutral classes (1:1). The designed experiment explores various H:F prior probability settings and their effects on performance. Table 15 lists these settings; they vary from a target dense environment (10:1) to a target sparse environment (1:10).

The last designed experimental parameter is the level of knowledge of the target initial aspect angle. Knowing *a priori* the target's pose relative to the sensor means searching a smaller template space for a match, thus reducing error. The initial setting was knowledge within $\pm 22.5^\circ$ of the targets true initial aspect angle. This assumption was verified by the pose estimation calculation of Sec. 5.4.5. The initial setting is supplemented with a degraded setting ($\pm 37.5^\circ$) and a setting where no prior knowledge of the target's initial aspect is used.

5.6.2.1 Prior knowledge of target aspect

This section discusses the impact of prior knowledge of target aspect based on results for the HMM-based and template-based classifier systems. Table 16 and Table 17 capture performance measures for an HMM-based system and a template-based system, respectively. The results are obtained from an experiment with co-located sensors, a medium observation length, and equal prior probabilities for hostile and friend/neutral classes.

Within the HMM-based results (Table 16) one sees a slight trend down in measures of robustness as the prior knowledge of target aspect worsens. The HMM-based system achieves jointly-feasible solutions in every case but for Sensor 2 acting independently in the $\pm 37.5^\circ$ case. For the template-based system (Table 17), jointly-feasible solutions exist in only 9 of 15 cases, and of these the number of jointly-feasible

Table 16. Performance comparison of prior aspect knowledge for HMM-based classifier, co-located sensors, medium observation length, and equal priors

Aspect	**Fusion**	**Percent feasible**						**Mean feasible value**						**Opt val**
		tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
								0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
±22.5°	Sensor 1	0.50	0.99	0.50	0.75	0.72	0.23	0.96	0.01	0.09	0.84	0.58	0.91	0.9723
	Sensor 2	0.50	0.99	0.32	0.75	0.75	0.07	0.94	0.04	0.04	0.84	0.65	0.91	0.9556
	Mean	0.50	0.99	0.50	0.75	0.75	0.25	0.96	0.02	0.09	0.84	0.64	0.91	0.9625
	ANN	0.42	0.79	0.37	0.73	0.70	0.06	0.99	0.03	0.05	0.78	0.62	0.98	1.0000
	Label	0.62	1.00	0.85	0.53	0.50	0.12	0.98	0.02	0.07	0.66	0.51	0.92	1.0000
±37.5°	Sensor 1	0.50	0.99	0.55	0.75	0.69	0.20	0.96	0.01	0.08	0.84	0.53	0.92	0.9855
	Sensor 2	0.32	0.99	0.55	0.75	0.68	-	0.98	0.04	0.06	0.85	0.57	-	-
	Mean	0.50	0.99	0.55	0.75	0.68	0.19	0.96	0.02	0.06	0.84	0.57	0.90	0.9014
	ANN	0.42	0.81	0.42	0.74	0.65	0.05	0.98	0.04	0.06	0.79	0.57	0.90	0.9567
	Label	0.56	1.00	0.80	0.53	0.46	0.09	0.97	0.01	0.05	0.66	0.47	0.89	0.9186
none	Sensor 1	0.50	0.99	0.50	0.75	0.75	0.25	0.96	0.04	0.10	0.84	0.56	0.92	0.9533
	Sensor 2	0.32	0.92	0.50	0.75	0.70	0.02	0.97	0.06	0.09	0.84	0.63	0.87	0.8773
	Mean	0.50	0.99	0.32	0.75	0.75	0.07	0.96	0.04	0.06	0.83	0.61	0.93	0.9499
	ANN	0.38	0.84	0.44	0.74	0.65	0.03	0.98	0.06	0.07	0.81	0.60	0.88	0.9057
	Label	0.62	1.00	0.85	0.39	0.45	0.10	0.98	0.02	0.07	0.66	0.47	0.91	0.9357

points is quite small (fewer than 5 points) versus the larger number for the HMM case (20 - 30 points).

Figure 63 singles out the case where Sensor 1 acts independently with no prior target aspect information. From Tables 16 and 17, one sees the share an optimal solution value of approximately 0.95, but the HMM-based classifier outperforms the template-based classifier when considering the size of the feasible regions. Focusing on the surface plots of Fig. 63, one sees that the two-tiered true-positive surface of the HMM-based classifier affords a larger feasible region (and larger jointly-feasible region) than the template-based classifier. The critical error surface in the HMM case is markedly better than the template case. Combined, the differences between the HMM and template case for the true-positive, critical error, and non-critical error feasible regions restrict the template-based jointly-feasible space to a single point while giving the HMM-based classifier a jointly-feasible region that is fully one-quarter of the threshold space.

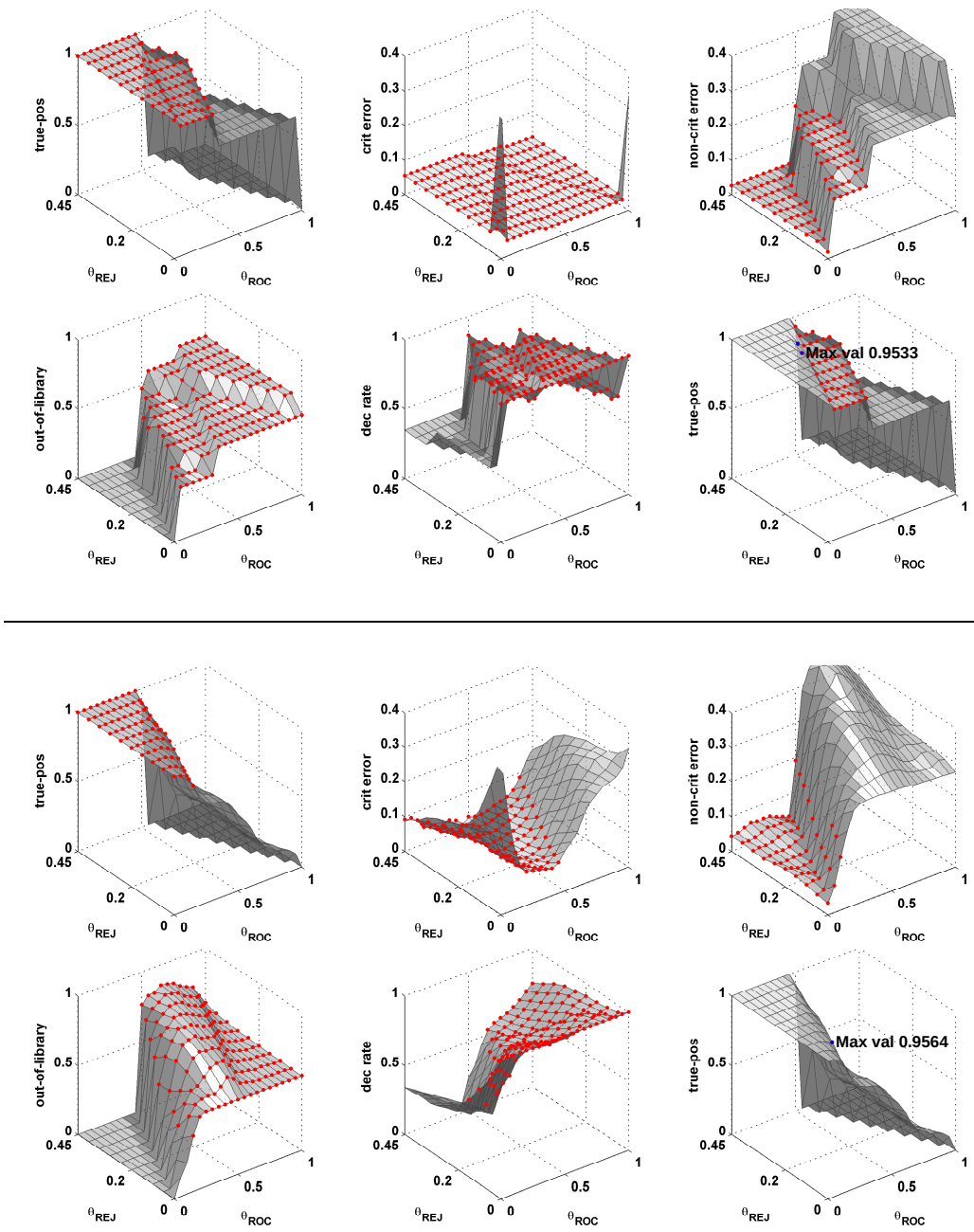

Figure 63. Performance surfaces determined by ROC threshold and reject threshold settings. HMM-based classifier surfaces are shown above the line, and template-based surfaces are shown below the line. Sensor 1 performance is with equal priors, medium observation length, and no prior knowledge of target aspect.

Table 17. Performance comparison of prior aspect knowledge for template-based classifier, co-located sensors, medium observation length, and equal priors

Aspect	**Fusion**	**Percent feasible**						**Mean feasible value**						**Opt val**
		tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
								0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
±22.5°	Sensor 1	0.36	0.92	0.40	0.75	0.60	-	0.98	0.05	0.07	0.83	0.48	-	-
	Sensor 2	0.36	0.87	0.40	0.75	0.65	0.01	0.98	0.05	0.06	0.82	0.53	0.87	0.8893
	Mean	0.38	0.95	0.40	0.75	0.68	0.06	0.98	0.03	0.06	0.83	0.56	0.91	0.9557
	ANN	0.37	0.51	0.35	0.68	0.67	-	0.98	0.06	0.06	0.81	0.58	-	-
	Label	0.40	0.98	0.69	0.43	0.20	-	0.99	0.03	0.06	0.67	0.38	-	-
±37.5°	Sensor 1	0.37	0.73	0.34	0.75	0.69	0.02	0.98	0.06	0.08	0.81	0.51	0.94	0.9532
	Sensor 2	0.35	0.83	0.40	0.73	0.66	0.02	0.98	0.06	0.06	0.83	0.56	0.87	0.8832
	Mean	0.38	0.80	0.37	0.75	0.68	0.04	0.98	0.04	0.07	0.81	0.55	0.93	0.9594
	ANN	0.38	0.59	0.34	0.70	0.69	0.00	0.99	0.06	0.08	0.79	0.67	0.97	0.9746
	Label	0.40	0.84	0.99	0.39	0.02	-	0.98	0.03	0.09	0.64	0.36	-	-
none	Sensor 1	0.36	0.54	0.32	0.71	0.70	0.00	0.98	0.05	0.07	0.80	0.65	0.96	0.9564
	Sensor 2	0.33	0.49	0.35	0.69	0.66	-	0.99	0.06	0.07	0.81	0.60	-	-
	Mean	0.36	0.59	0.34	0.70	0.69	0.00	0.98	0.05	0.06	0.81	0.62	0.95	0.9450
	ANN	0.35	0.74	0.39	0.71	0.62	-	0.98	0.07	0.08	0.80	0.53	-	-
	Label	0.38	0.77	0.73	0.31	0.43	0.00	0.99	0.03	0.07	0.64	0.48	0.88	0.9363

5.6.2.2 Target class prior probabilities

The designed experiment explores the effects of target class prior probabilities across a range of settings from target dense (10:1) to target sparse (1:10). Table 18 shows comparative results for an HMM-based system and a template-based system using a single sensor (Sensor 1) at the long observation length setting with prior knowledge of the target aspect to within ±22.5°. Results span the settings for target class prior probabilities.

One notices that the template-based system is relatively robust to changes in the prior probabilities of the target classes. The HMM-based classifier performs well in the target rich environment, but there is a break point moving from equal priors (1:1) to target sparse (1:2) and beyond, where the non-critical feasibility space shrinks by half and eliminates all jointly-feasible solutions.

Table 18. Performance comparisons across target class prior probability settings: Sensor 1, long observation length, and ±22.5° target aspect knowledge

	Priors	**Percent feasible**						**Mean feasible value**						**Opt val**
	(H:F)	tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
								0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
HMM	10:1	0.50	1.00	0.50	0.74	0.75	0.25	0.97	0.01	0.04	0.83	0.66	0.94	0.9884
	4:1	0.50	0.99	0.50	0.74	0.75	0.25	0.97	0.01	0.06	0.81	0.66	0.94	0.9884
	2:1	0.50	0.99	0.50	0.78	0.75	0.25	0.97	0.01	0.07	0.80	0.66	0.94	0.9884
	1:1	0.50	0.99	0.50	0.75	0.75	0.25	0.97	0.00	0.09	0.84	0.66	0.94	0.9884
	1:2	0.50	0.99	0.25	0.75	0.75	-	0.97	0.00	0.01	0.89	0.66	-	-
	1:4	0.50	0.99	0.25	0.75	0.75	-	0.97	0.00	0.01	0.93	0.66	-	-
	1:10	0.50	1.00	0.25	0.75	0.75	-	0.97	0.01	0.01	0.96	0.66	-	-
Template	10:1	0.36	0.85	0.39	0.76	0.65	0.01	0.99	0.03	0.06	0.75	0.53	0.95	0.9500
	4:1	0.36	0.89	0.39	0.76	0.65	0.02	0.99	0.04	0.06	0.77	0.53	0.94	0.9500
	2:1	0.36	0.92	0.40	0.76	0.65	0.02	0.99	0.04	0.06	0.79	0.53	0.94	0.9500
	1:1	0.36	0.91	0.40	0.75	0.65	0.02	0.99	0.04	0.06	0.82	0.53	0.94	0.9500
	1:2	0.36	0.88	0.40	0.75	0.65	0.02	0.99	0.03	0.06	0.86	0.53	0.94	0.9500
	1:4	0.36	0.77	0.41	0.74	0.65	0.02	0.99	0.02	0.07	0.89	0.53	0.94	0.9500
	1:10	0.36	0.72	0.42	0.74	0.65	0.02	0.99	0.02	0.07	0.92	0.53	0.94	0.9500

Figure 64 shows the performance surfaces for target rich (10:1) and target sparse (1:10) environments for the HMM-based single sensor system operating with long observation sequences and ±22.5° target aspect knowledge. One sees the encroachment of the infeasible portion of the non-critical error surface as the target priors shift from 10:1 to 1:10. The combination of decreased declaration performance and infeasible non-critical error surface removes all jointly-feasible solutions.

Relative changes in target class prior probabilities do not effect performance measures based on horizontal analysis of confusion matrices. Horizontal analysis focuses on the number of true class records that are correctly labeled given that a declaration is made. Thus, in an initial experiment with equal class prior probabilities and a classifier estimated true-positive performance for a given target class i of 90%, then 90 of 100 class i records are correctly labeled given a declaration is made on every record. In a follow-up experiment using a target sparse class prior probability of 1:10, 9 of 10 records from class i are correctly labeled. In either case the estimated true-positive performance for class i for the classifier is 90%. Evidence

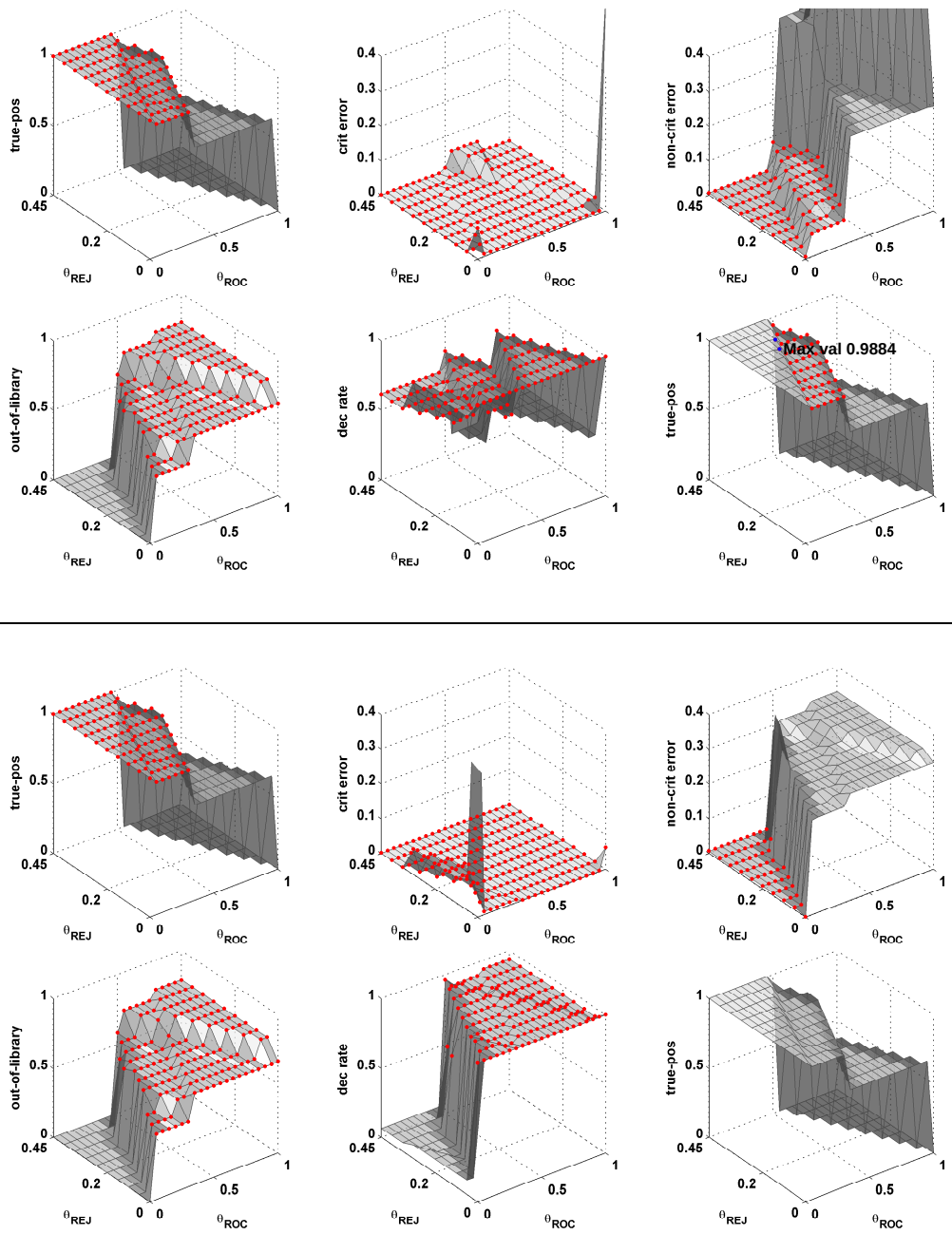

Figure 64. Performance surfaces determined by ROC threshold and reject threshold settings. HMM-based classifier surfaces for a target dense environment (10:1) are above the line; surfaces for a target sparse environment (1:10) are below the line. Sensor 1 performance, long observation length, and ±22.5° target aspect knowledge are used.

of this result can be seen in the top and bottom subplots of Fig. 64 for true-positive, out-of-library, and jointly-feasible (true-positive) performance surfaces. Whether the class prior probabilities are 10:1 (above the line) or 1:10 (below the line), the surface shapes do not change for these horizontal analysis performance measures.

Vertical analysis of the confusion matrix measures the probability of correct labeling, or how often correct given that a class i label is applied by the classifier. The critical error, non-critical error, and declaration performance measures use vertical analysis of the confusion matrices and are influenced by the class prior probabilities. Evidence of this influence can be seen in the different surface shapes for these performance measures in Fig. 64.

5.6.2.3 Observation sequence length

The designed experiment explores the effects of changing observation sequence length. The initial experiment uses a medium observation sequence length setting of 5° of target azimuth. A short observation length setting uses 2° of target azimuth and a long setting uses 10°.

Table 19 shows performance results for the various observation length settings for an HMM-based system with co-located sensors, equal class prior probabilities, and no prior target aspect knowledge. Table 20 shows results at the same settings but for a template-based CID system.

Table 19 shows a general improvement in robustness with increased observation sequence length. This result can be seen in the percentage of the threshold space that is jointly-feasible. For the short observation length setting, only the boolean fusion method shows any jointly-feasible settings. At the medium setting, all methods show jointly-feasible solutions. At the long setting, the size of the jointly-feasible space increases (except for Sensor 2, which suffered a decrease in non-critical feasibility), with best performance occurring for the boolean fusion method.

Table 19. Performance comparison of observation sequence lengths for HMM-based classifier, co-located sensors, equal priors, and no prior target aspect knowledge

Obs Length	**Fusion**	**Percent feasible**						**Mean feasible value**						**Opt val**
		tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
								0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
Short	Sensor 1	0.44	0.99	0.31	0.76	0.68	-	0.95	0.07	0.08	0.83	0.55	-	-
	Sensor 2	0.26	-	0.26	0.75	0.74	-	1.00	-	0.03	0.84	0.54	-	-
	Mean	0.50	0.74	0.25	0.78	0.75	-	0.96	0.07	0.02	0.81	0.51	-	-
	ANN	0.35	0.72	0.42	0.71	0.56	-	0.98	0.06	0.08	0.82	0.45	-	-
	Label	0.60	0.97	0.62	0.39	0.51	0.11	0.97	0.06	0.07	0.67	0.46	0.88	0.9431
Medium	Sensor 1	0.50	0.99	0.50	0.75	0.75	0.25	0.96	0.04	0.10	0.84	0.56	0.92	0.9533
	Sensor 2	0.32	0.92	0.50	0.75	0.70	0.02	0.97	0.06	0.09	0.84	0.63	0.87	0.8773
	Mean	0.50	0.99	0.32	0.75	0.75	0.07	0.96	0.04	0.06	0.83	0.61	0.93	0.9499
	ANN	0.38	0.84	0.44	0.74	0.65	0.03	0.98	0.06	0.07	0.81	0.60	0.88	0.9057
	Label	0.62	1.00	0.85	0.39	0.45	0.10	0.98	0.02	0.07	0.66	0.47	0.91	0.9357
Long	Sensor 1	0.50	0.99	0.55	0.75	0.75	0.25	0.95	0.02	0.08	0.84	0.65	0.90	0.9342
	Sensor 2	0.26	0.99	0.25	0.75	0.75	-	1.00	0.06	0.01	0.85	0.61	-	-
	Mean	0.50	0.99	0.50	0.75	0.75	0.25	0.94	0.03	0.08	0.84	0.71	0.89	0.9394
	ANN	0.39	0.71	0.47	0.74	0.64	0.02	0.98	0.05	0.07	0.81	0.54	0.91	0.9129
	Label	0.56	0.99	0.85	0.53	0.54	0.17	0.96	0.02	0.07	0.64	0.45	0.89	0.9652

Table 20. Performance comparison of observation sequence lengths for template-based classifier, co-located sensors, equal priors, and no prior target aspect knowledge

Obs Length	**Fusion**	**Percent feasible**						**Mean feasible value**						**Opt val**
		tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
								0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
Short	Sensor 1	0.32	0.37	0.29	0.70	0.70	-	0.99	0.07	0.07	0.80	0.53	-	-
	Sensor 2	0.31	0.29	0.32	0.70	0.68	-	0.99	0.07	0.07	0.79	0.53	-	-
	Mean	0.32	0.49	0.30	0.70	0.69	-	0.99	0.07	0.07	0.79	0.54	-	-
	ANN	0.34	0.21	0.37	0.67	0.63	-	0.98	0.08	0.09	0.82	0.54	-	-
	Label	0.34	0.59	0.67	0.29	0.36	0.00	0.99	0.05	0.09	0.63	0.48	0.89	0.9246
Medium	Sensor 1	0.36	0.54	0.32	0.71	0.70	0.00	0.98	0.05	0.07	0.80	0.65	0.96	0.9564
	Sensor 2	0.33	0.49	0.35	0.69	0.66	-	0.99	0.06	0.07	0.81	0.60	-	-
	Mean	0.36	0.59	0.34	0.70	0.69	0.00	0.98	0.05	0.06	0.81	0.62	0.95	0.9450
	ANN	0.35	0.74	0.39	0.71	0.62	-	0.98	0.07	0.08	0.80	0.53	-	-
	Label	0.38	0.77	0.73	0.31	0.43	0.00	0.99	0.03	0.07	0.64	0.48	0.88	0.9363
Long	Sensor 1	0.36	0.54	0.37	0.69	0.63	-	0.99	0.05	0.09	0.83	0.52	-	-
	Sensor 2	0.35	0.43	0.31	0.70	0.67	-	0.98	0.06	0.10	0.79	0.54	-	-
	Mean	0.38	0.52	0.30	0.71	0.69	-	0.98	0.04	0.06	0.78	0.58	-	-
	ANN	0.35	0.57	0.44	0.71	0.60	-	0.98	0.08	0.09	0.82	0.53	-	-
	Label	0.39	0.75	0.85	0.31	0.18	-	0.98	0.03	0.09	0.65	0.38	-	-

Table 20 shows a general improvement in robustness from short to medium observation sequence length. Notably, the critical error feasibility space improves and the out-of-library performance improves, yielding jointly-feasible solutions for three of the five fusion methods.

At the long observation sequence length setting, the template-based classifier performance fell slightly in several areas, reducing the feasible space in those performance categories and removing all jointly-feasible solutions. The additional information gained from a longer observation sequence did not benefit the template-based classifier. The mean vector and covariance matrix employed in the template-based classification methodology may have been corrupted by additional noise contained in the longer observation sequences.

Figure 65 shows the performance surfaces for the cases of short observation length and long observation length for the HMM-based co-located sensors using the mean fusion method, equal class prior probabilities, and no prior target aspect knowledge. Marked improvement in the critical error, non-critical error, and out-of-library surfaces is evident.

The critical error surface drops several percentage points and becomes almost entirely feasible. The non-critical error surface drops and increases its feasible region. Non-critical error is the limiting factor for the short observation length lack of jointly-feasible solutions. The improvement in non-critical feasibility creates a sizable jointly-feasible space at the long observation length. Out-of-library performance improves, but not the size of the feasible region.

5.6.2.4 Sensor correlation

The last area explored by the designed experiment is the location of sensors. Two settings are used; the first has the sensors co-located on a single platform, the second has the sensors located on separate platforms. The co-located sensors produce

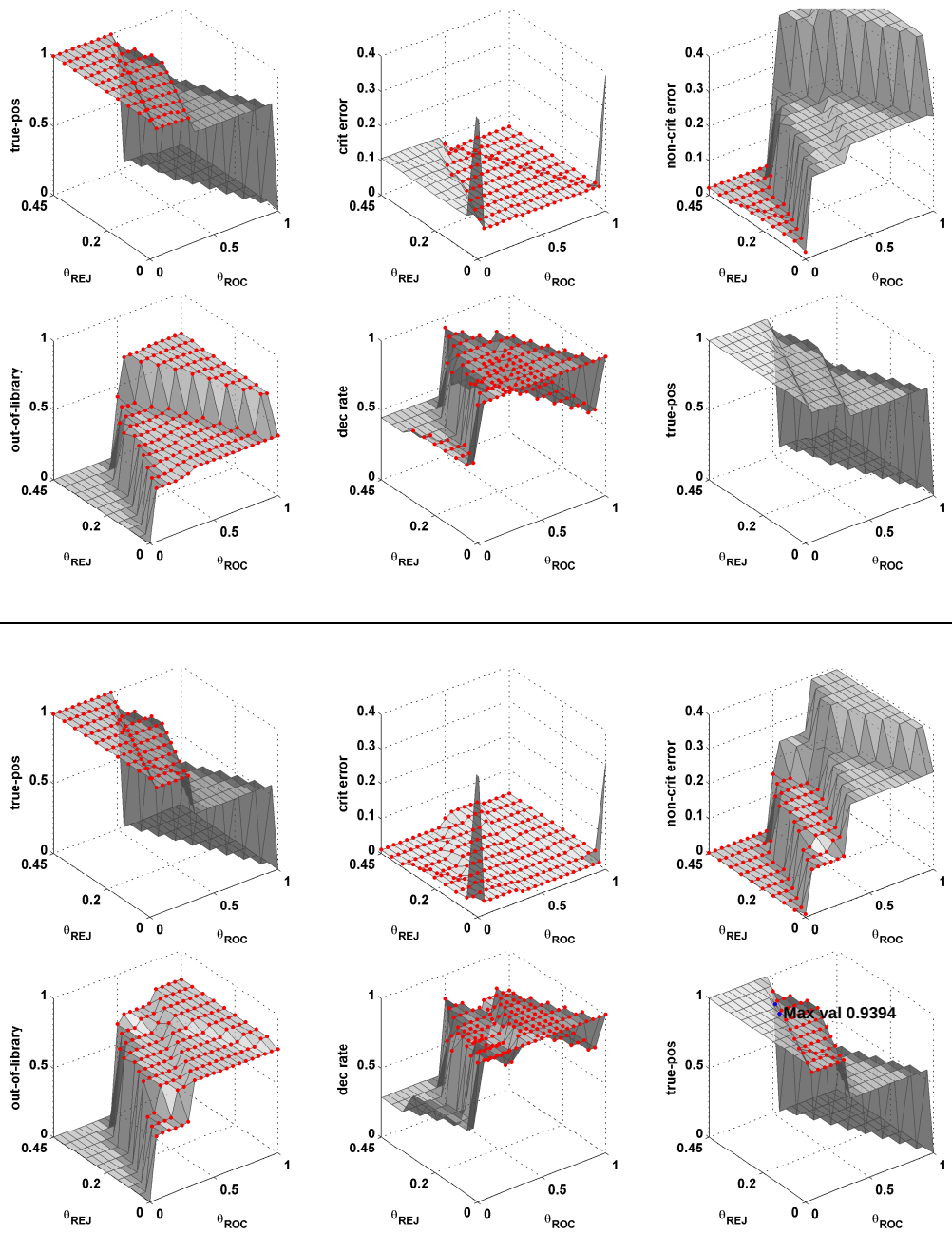

Figure 65. Performance surfaces determined by ROC threshold and reject threshold settings. HMM-based classifier surfaces for short observation lengths are above the line, and surfaces for long observation lengths are shown below the line. Co-located sensors, mean fusion method, equal target class priors, and no prior target aspect knowledge are used.

Table 21. Performance comparison of sensor location for template-based classifier, equal priors, long observation length, and no prior target aspect knowledge

Sensors	**Fusion**	**Percent feasible**						**Mean feasible value**						**Opt val**
		tp	crit	n-crit	dec	ool	joint	tp	crit	n-crit	dec	ool	joint	
								0.85	**0.1**	**0.2**	**0.5**	**0.35**	**0.85**	
Co-located	Sensor 1	0.36	0.54	0.37	0.69	0.63	-	0.99	0.05	0.09	0.83	0.52	-	-
	Sensor 2	0.35	0.43	0.31	0.70	0.67	-	0.98	0.06	0.10	0.79	0.54	-	-
	Mean	0.38	0.52	0.30	0.71	0.69	-	0.98	0.04	0.06	0.78	0.58	-	-
	ANN	0.35	0.57	0.44	0.71	0.60	-	0.98	0.08	0.09	0.82	0.53	-	-
	Label	0.39	0.75	0.85	0.31	0.18	-	0.98	0.03	0.09	0.65	0.38	-	-
Independent	Sensor 1	0.37	0.52	0.33	0.71	0.70	0.01	0.98	0.06	0.06	0.80	0.61	0.96	0.9644
	Sensor 2	0.34	0.46	0.29	0.71	0.70	-	0.98	0.06	0.09	0.78	0.57	-	-
	Mean	0.36	0.63	0.34	0.70	0.70	0.01	0.98	0.05	0.08	0.80	0.60	0.96	0.9571
	ANN	0.35	0.62	0.30	0.73	0.72	0.01	0.98	0.07	0.06	0.79	0.64	0.94	0.9433
	Label	0.39	0.80	0.67	0.26	0.39	0.01	0.99	0.03	0.07	0.63	0.47	0.91	0.9831

observations that begin and end at shared target azimuth points. The independent sensors produce observation sequences of the same aspect window size but that begin and end at different target azimuth points (see Fig. 62).

Table 21 shows performance results for the two sensor location settings for a template-based system with equal class prior probabilities, long observation length, and no prior target aspect knowledge.

Table 21 indicates an improvement in Sensor 1 performance which leads to jointly-feasible solutions for the fused methods (mean, ANN, and boolean) in the independent sensor case. The slight improvement in the size of the declaration and out-of-library feasible spaces for Sensor 2 translates into benefits when fused with Sensor 1. Performance gains can be attributed to the additional information resulting from viewing the target from two different orientations (independent sensors) versus a single orientation (co-located platform). If a single platform presents the target at an orientation where poor target discrimination information exists, then co-located sensor performance is poor. Given the same orientation for one of two independent

sensors, the second orientation may mitigate the poor performance by supplementing information with greater discriminatory power from the second orientation.

Figure 66 shows the performance surfaces for the cases of co-located sensors and independent sensors for the HMM-based system using the mean fusion method, equal class prior probabilities, no prior target aspect knowledge, and long observation sequence length.

Slight improvements in critical error, non-critical error, and out-of-library feasibility spaces lead to several jointly-feasible solutions. The additional information provided by independent sensors results in slight enhancement of the fused feasible regions, leading to jointly-feasible solutions where none existed for the co-located sensor arrangement.

5.6.2.5 Label fusion sensor thresholds

The surface plots used in the previous sections showed performance of all of the fusion methods except label fusion. The label fusion methodology used in both the initial and designed experiments employs a set of ROC and rejection region thresholds for each classifier. The labeling of test records occurs at each classifier, whereupon the labels are fused via a set of label fusion rules. In the case of the other fusion methods, classifier outputs are fused and then labeled with a single application of the ROC and rejection region thresholds. Thus, the label fusion method has a four-dimensional threshold space, versus the two-dimensional space of the other fusion methods.

Figure 67 shows the location in threshold space for classifiers 1 and 2 of the optimal solution(s) when the label fusion rule is applied in the case of co-located sensors with an HMM-based system.

For the target dense environments (10:1, 4:1, 2:1, and 1:1), the optimal threshold settings remain the same. Classifier 2 thresholds are held at two locations, while

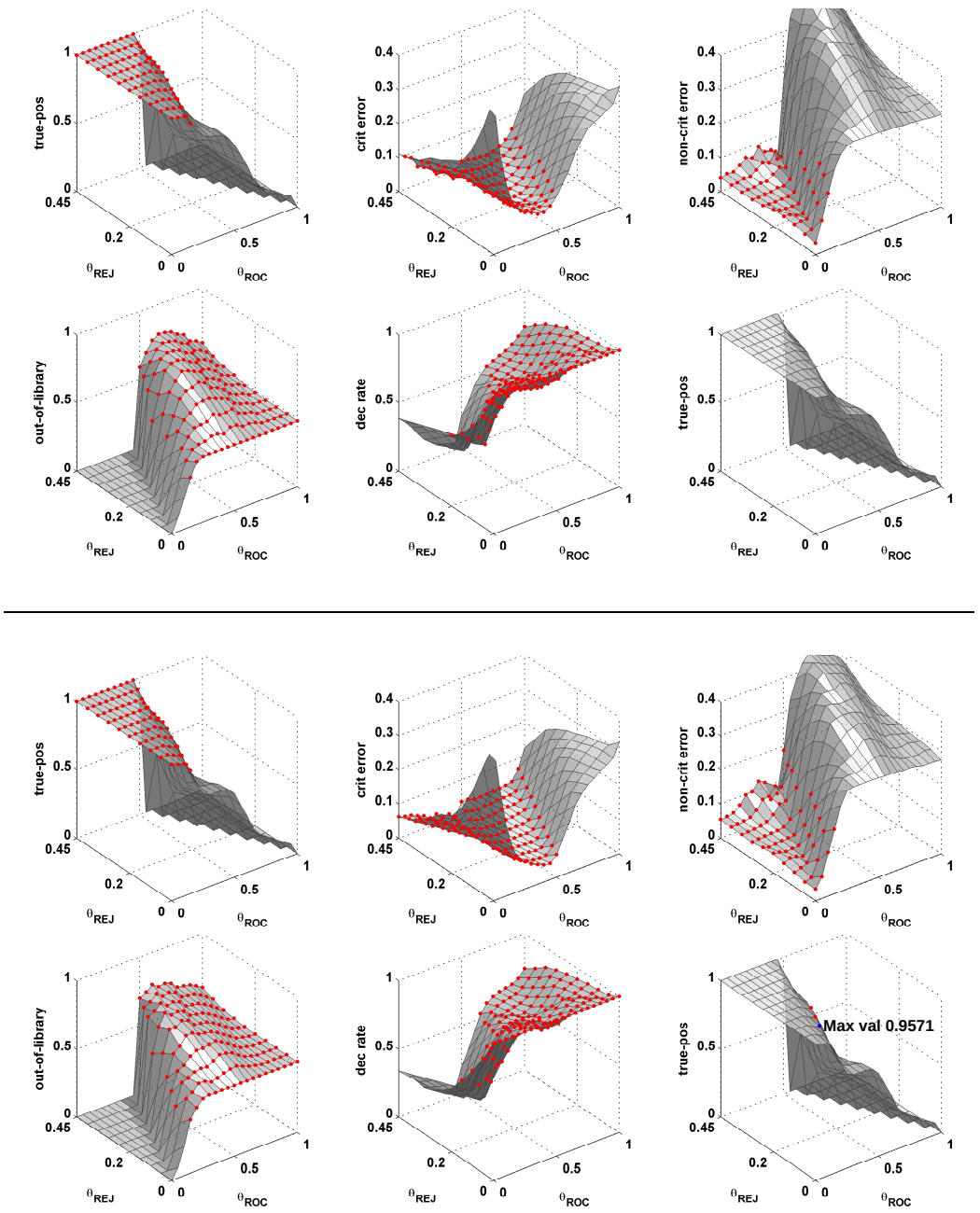

Figure 66. Performance surfaces determined by ROC threshold and reject threshold settings. Template-based classifier surfaces for co-located sensors are above the line, and surfaces for independent sensors are below the line. Mean fusion method, equal target class priors, no prior target aspect knowledge, and long observation length are used.

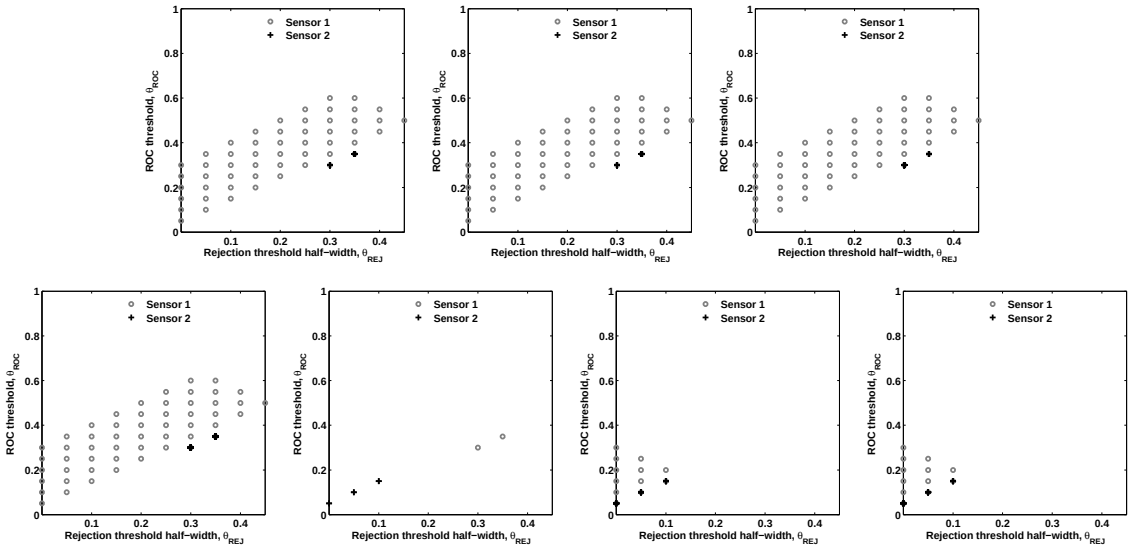

Figure 67. Optimal threshold settings for classifiers 1 and 2 when label fusion is applied. Subplots correspond to class prevalence settings for hostile to friend/neutral ratios of 10:1, 4:1, 2:1, 1:1, 1:2, 1:4, and 1:10 respectively. A HMM-based system with co-located sensors, long observation length, and prior target aspect $\pm 22.5^\circ$ is used.

classifier 1 thresholds vary widely in the threshold space. In the target sparse environments the optimal threshold settings change. Indeed, in the case where hostile to friend/neutral prior probability is 1:2, the optimal threshold locations for classifiers 1 and 2 separate, indicating the label fusion rule's flexibility allows each classifier to perform well in a different area.

Figure 68 shows optimal threshold settings for the case where there is no prior target aspect knowledge. Behavior similar to Fig. 67 is seen, where optimal threshold settings for each classifier occur at different locations for the target dense settings. A much smaller jointly-optimal solution space exists due to reduced classifier performance resulting from lack of prior target aspect information.

Figure 69 shows optimal threshold settings for the case where classifiers 1 and 2 are located on different platforms. Classifiers 1 and 2 are combined with template-based classifiers acting on long observation sequences with prior target aspect knowledge of $\pm 37.5^\circ$. Again, the optimal threshold settings change as the

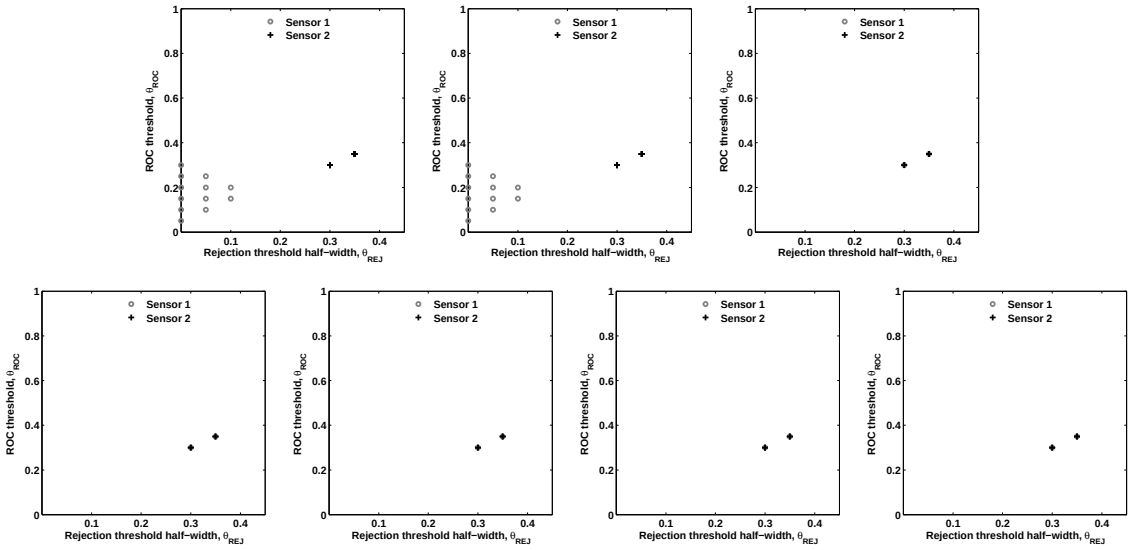

Figure 68. Optimal threshold settings for sensors 1 and 2 when label fusion is applied. Subplots correspond to class prevalence settings for hostile to friend/neutral ratios of 10:1, 4:1, 2:1, 1:1, 1:2, 1:4, and 1:10, respectively. A HMM-based system with co-located sensors, long observation length and no prior target aspect knowledge is used.

hostile to friend/neutral prior probability ratio changes. Also, optimal threshold settings for classifier 1 and classifier 2 occur in different locations.

The label fusion rule provides the multiple classifier system with greater flexibility in setting its thresholds compared to the other fusion methods. This flexibility allows the system to optimize sensor performance independently. The information lost in using a relatively simple set of label fusion rules is compensated by the flexibility inherent in the larger threshold space. As a result, label fusion performance is comparable to that of the other fusion methods.

5.6.2.6 Combined results

Figures 70- 75 provide performance results across all settings within the designed experiment. The performance results are presented using grayscale: lighter shades are better, white is best, and black is worst.

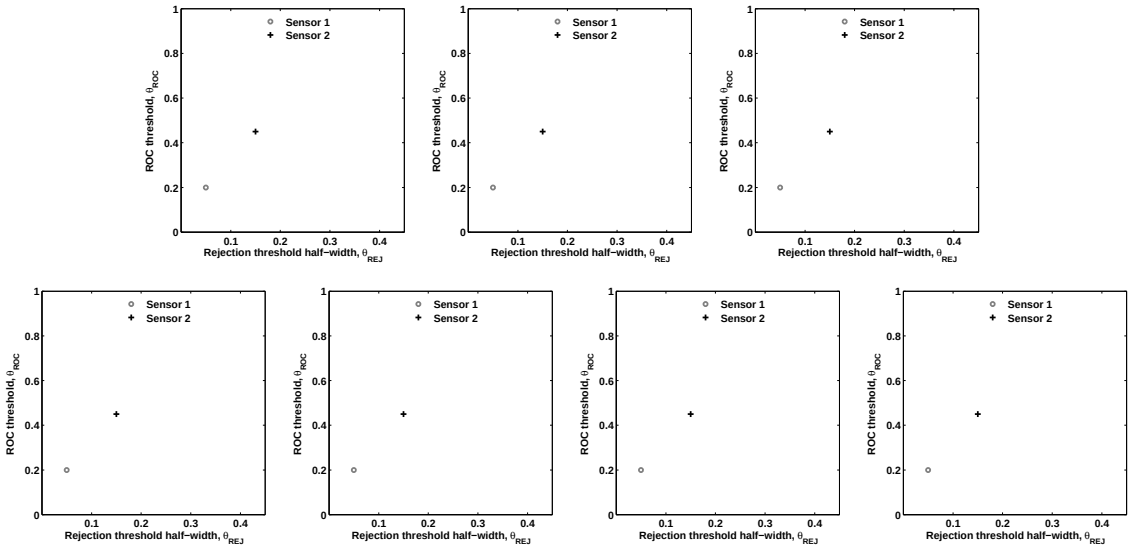

Figure 69. Optimal threshold settings for classifiers 1 and 2 when label fusion is applied. Subplots correspond to class prevalence settings for hostile to friend/neutral ratios of 10:1, 4:1, 2:1, 1:1, 1:2, 1:4, and 1:10, respectively. A template-based system with independent sensors, long observation length and prior target aspect $\pm 37.5^{\circ}$ is used.

A single figure has two subplots. The top subplot shows performance results for an HMM-based system. The bottom shows results from a template-based system. Each pair of subplots show results from the same designed experimental settings for sensor location and prior knowledge of target aspect. Performance results within each subplot explore experimental settings of observation length, fusion method, and target class prior probabilities.

Using a single subplot as an example and starting down the left-hand side, five different fusion methods are apparent: Sensor 1 acting independently, Sensor 2 acting independently, Sensors 1 and 2 fused with a mean fusion rule, Sensors 1 and 2 fused with a neural network fusion rule, and Sensors 1 and 2 fused using label fusion. Within each of these fusion categories, performance is given by target class prior probabilities, which are listed on the right-hand side of the subplot and cover the settings 10:1, 4:1, 2:1, 1:1, 1:2, 1:4, and 1:10 for hostile to friend/neutral target class ratios.

Across the top of the subplot are the three observation length settings: short, medium, and long. Within each observation length setting performance is captured using the same criteria as the tables used earlier. These criteria include measures of robustness for each performance category (labeled "robust" at the bottom), the mean feasible values for each performance category, and the optimal jointly-feasible value. The performance categories include true-positive, critical error, non-critical error, declaration, out-of-library, and joint performance. These categories are repeated for each observation length setting.

Figure 70 shows results for HMM-based (top) and template-based (bottom) systems where the sensors are co-located and prior knowledge of target aspect is $\pm 22.5°$. The best performance at this setting is the HMM-based system with neural network fusion. As observation length increases (reading from left to right), jointly-feasible performance improves, attaining optimal solutions for nearly all class prevalence settings.

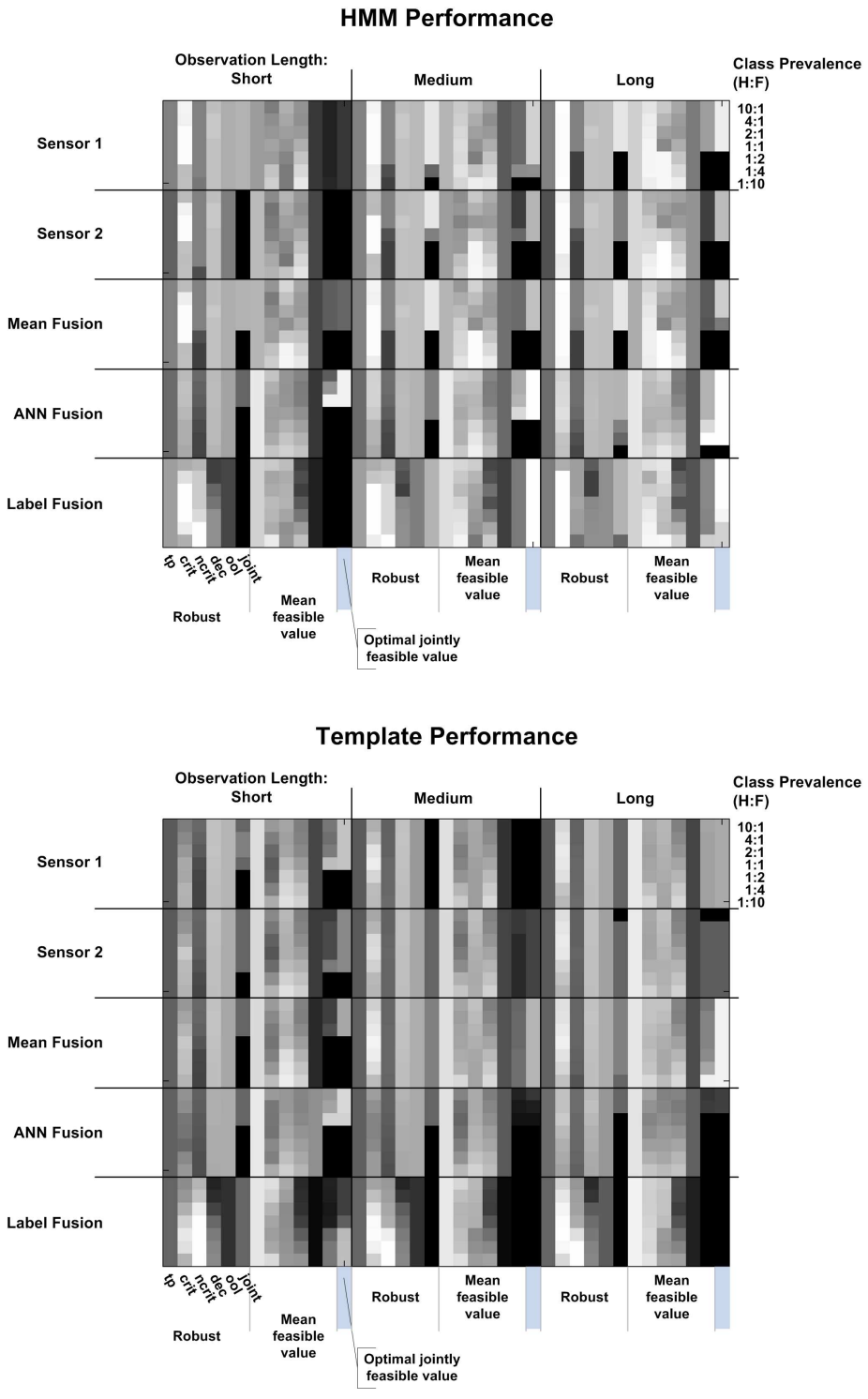

Figure 70. Combined results for HMM-based (top) and template-based (bottom) classifiers with co-located sensors and $\pm 22.5^\circ$ prior target aspect knowledge

Figure 71 shows results for HMM-based (top) and template-based (bottom) systems where the sensors are co-located and prior knowledge of target aspect is $\pm 37.5°$. The degraded prior aspect knowledge affects the HMM-based system at short observation lengths as evidenced by the large black jointly-feasible regions for Sensor 1, Sensor 2, and label fusion. As observation length increases, the HMM-based classifier system overcomes the degraded prior target aspect knowledge. With neural network fusion at the longest observation length setting, performance is near perfect at all but two class prevalence settings.

The template-based system fared better than the HMM system at the short observation length setting. However, at the long observation length setting only a few combinations yield jointly-feasible solutions. In both the HMM and template cases the neural network fusion method outperformed its competitors.

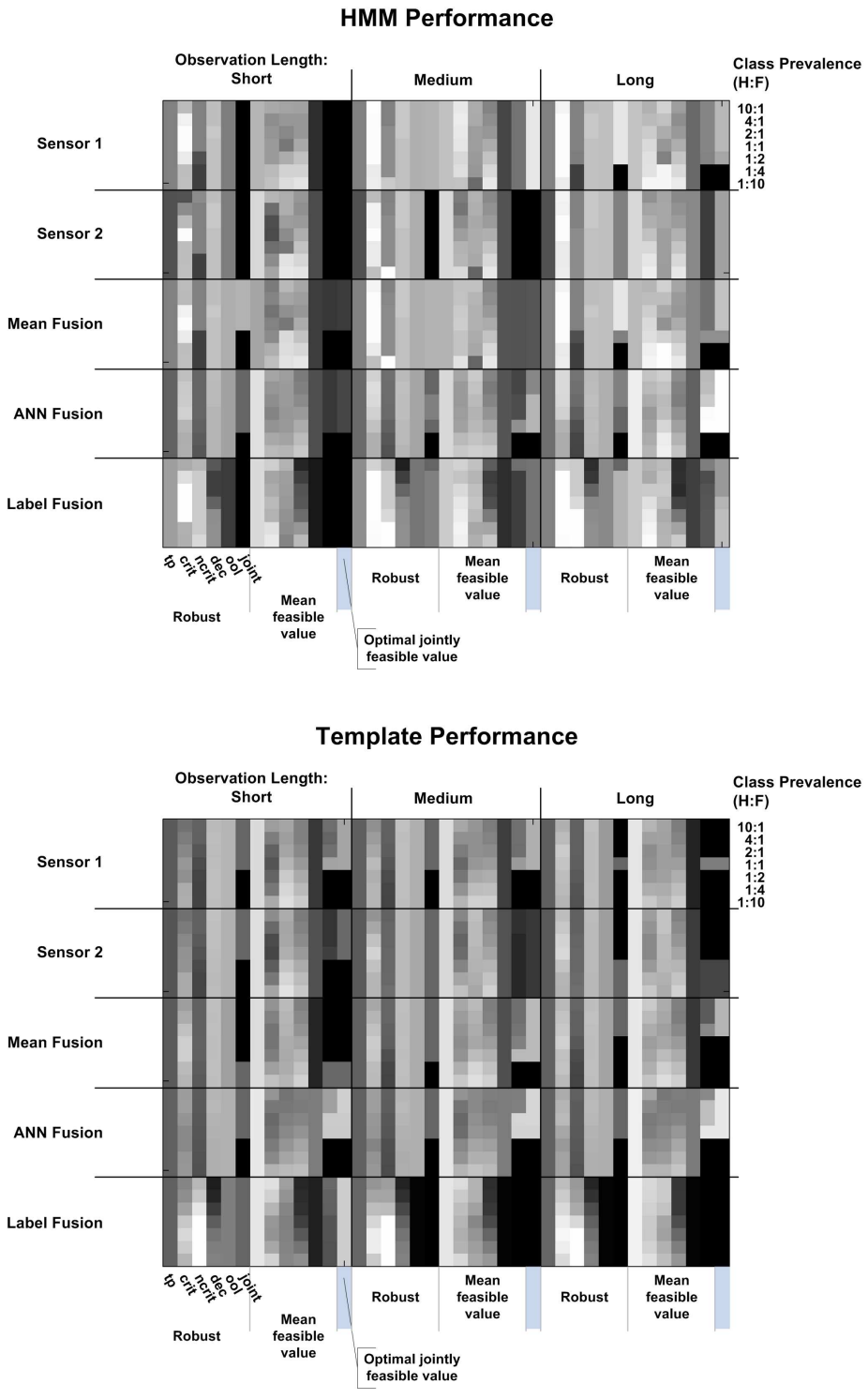

Figure 71. Combined results for HMM-based (top) and template-based (bottom) classifiers with co-located sensors and $\pm 37.5^\circ$ prior target aspect knowledge

Figure 72 shows results for HMM-based (top) and template-based (bottom) systems where the sensors are co-located with no prior knowledge of target aspect. The degraded prior aspect knowledge affects the template-based classifier system more dramatically than in Fig. 71. The HMM-based system provides good performance at the medium and long observation length settings with label fusion proving to be best.

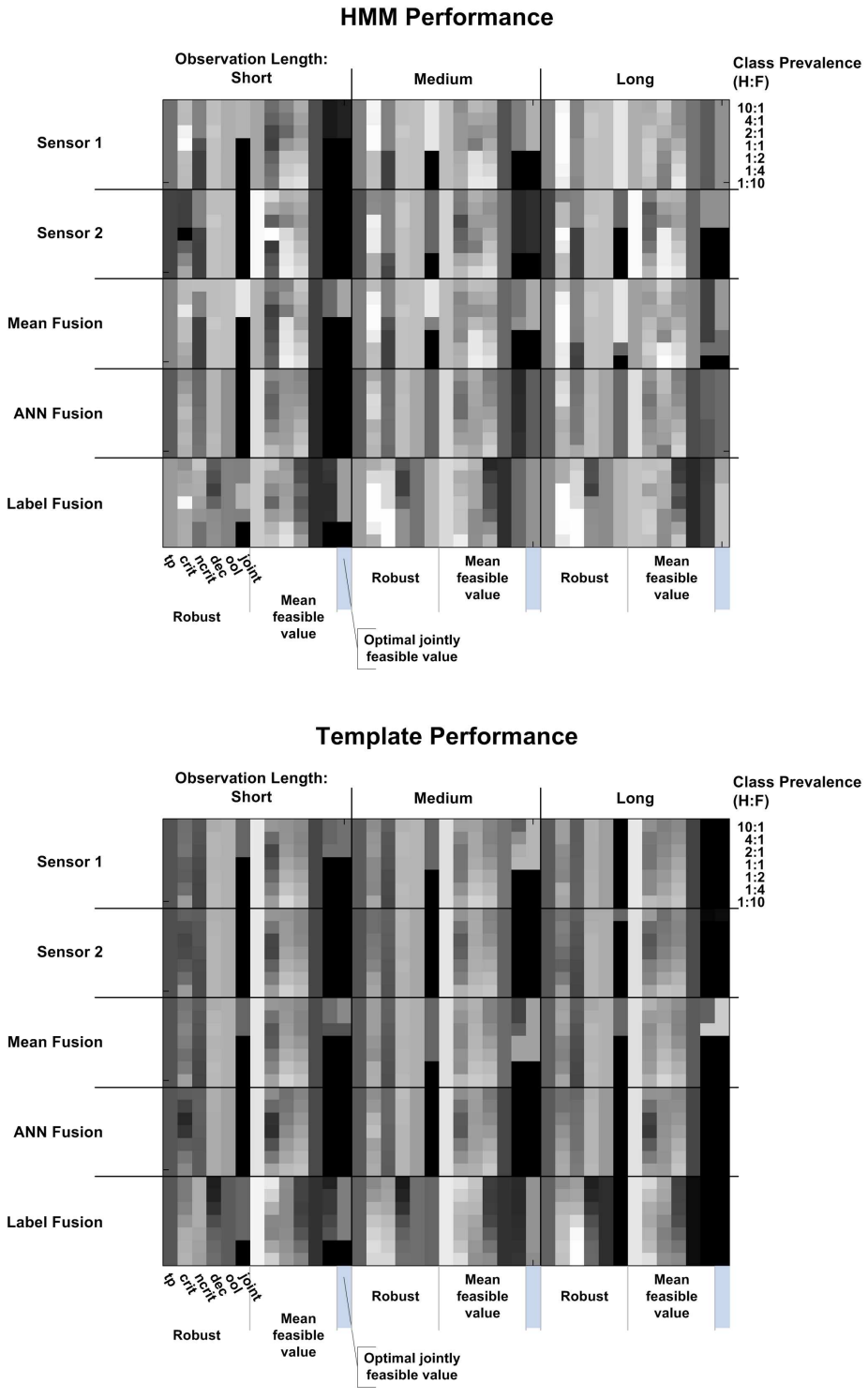

Figure 72. Combined results for HMM-based (top) and template-based (bottom) classifiers with co-located sensors and no prior target aspect knowledge

Figure 73 shows results for HMM-based (top) and template-based (bottom) systems where the sensors are located on different platforms with prior knowledge of target aspect $\pm 22.5°$. Independent sensors improve feasibility at the short and medium observation length settings for the HMM-based system compared to Fig. 70, but reduced feasibility for the neural network fusion method, which had been the best performer.

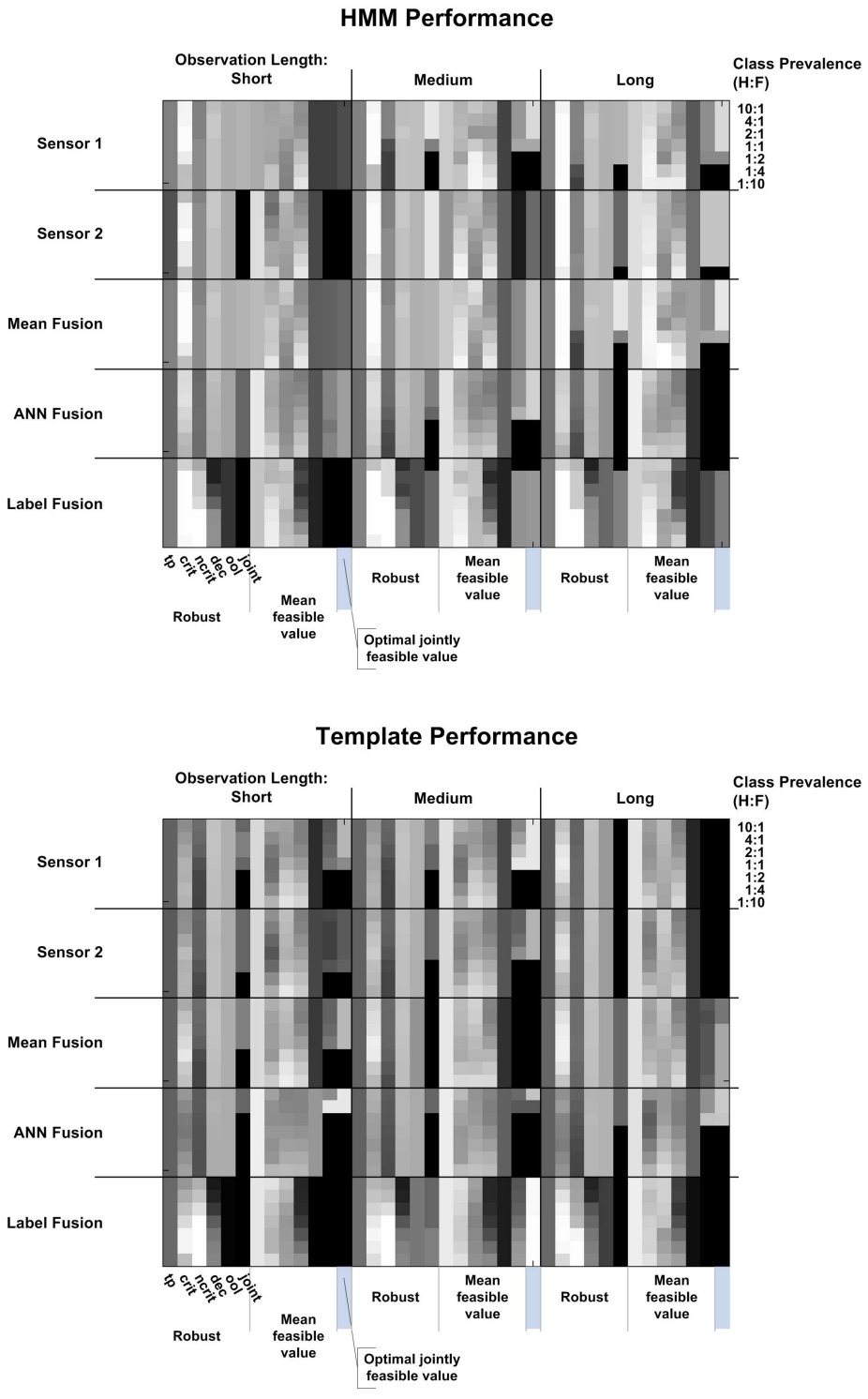

Figure 73. Combined results for HMM-based (top) and template-based (bottom) classifiers with independent sensors and $\pm 22.5^\circ$ prior target aspect knowledge

Figure 74 shows results for HMM-based (top) and template-based (bottom) systems where the sensors are located on different platforms with prior knowledge of target aspect $\pm 37.5^{\circ}$. Independent sensor improved performance levels over those shown in Fig. 71, but no significant changes in feasibility are observed.

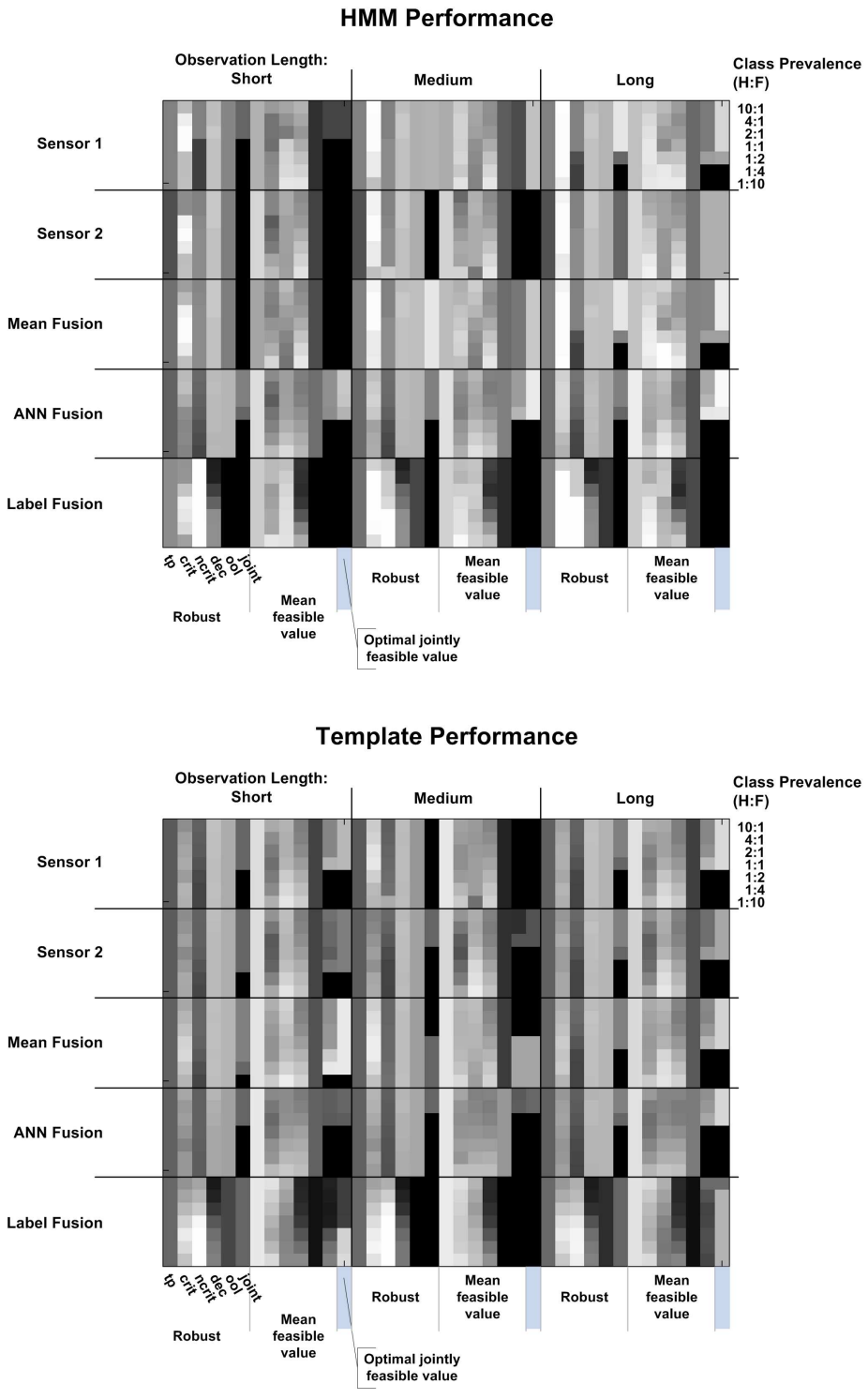

Figure 74. Combined results for HMM-based (top) and template-based (bottom) classifiers with independent sensors and $\pm 37.5^\circ$ prior target aspect knowledge

Figure 75 shows results for HMM-based (top) and template-based (bottom) systems where the sensors are located on different platforms with no prior knowledge of target aspect. Independent sensor improved performance levels over those shown in Fig. 72. Significant improvement in jointly-feasible space occurred in the template-based system at the long observation length setting

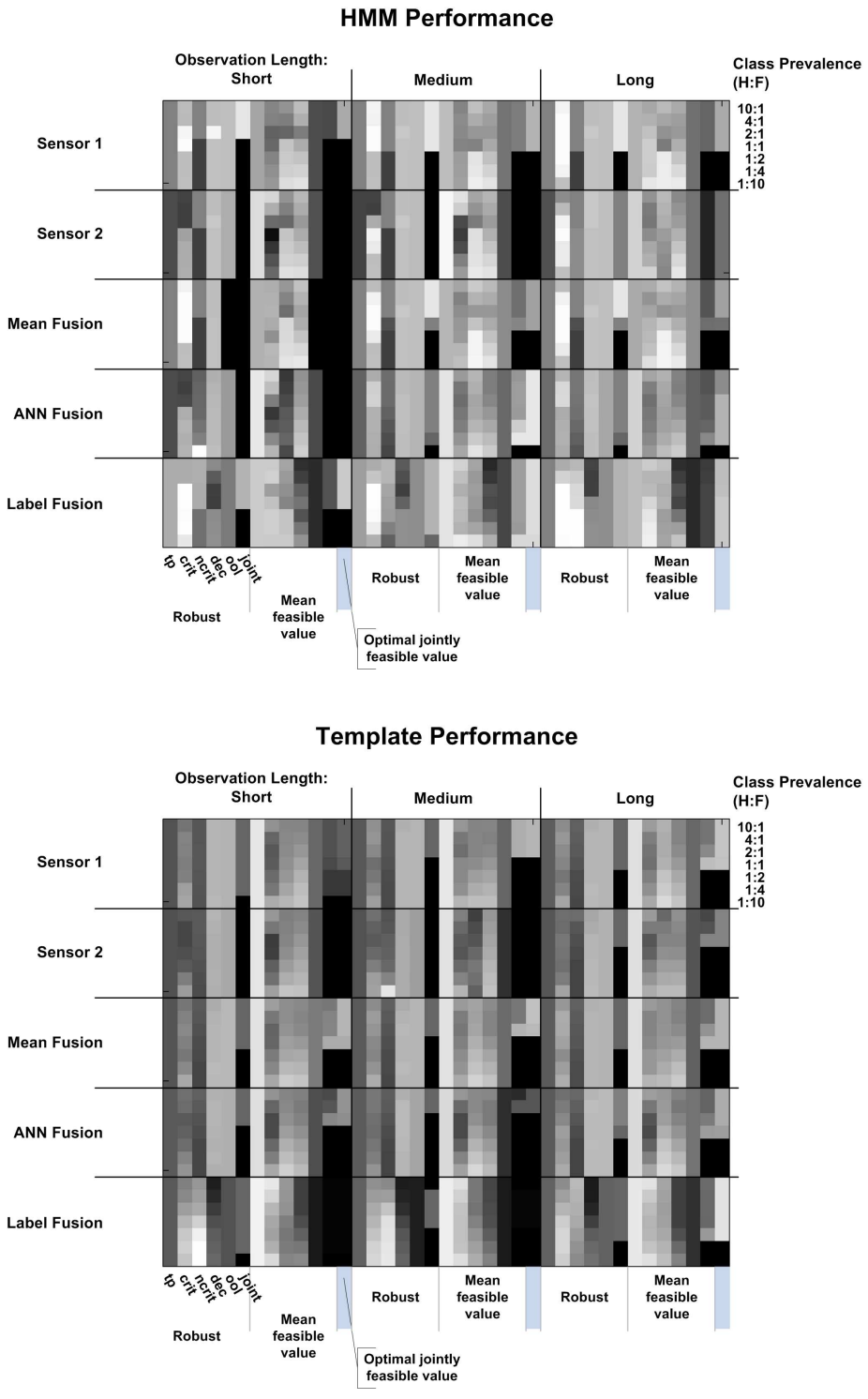

Figure 75. Combined results for HMM-based (top) and template-based (bottom) classifiers with independent sensors and no prior target aspect knowledge

A summary of the combined results notes the broad advantage in robustness and performance measure values held by the HMM-based system over the template-based system across experimental settings. The combination of HMM-based classifiers and neural fusion produced the best results when sensors were co-located and with some prior knowledge of target aspect available. With no prior aspect knowledge the boolean fusion method performed best: its ability to set sensor thresholds independently overcame its simple set of label fusion rules. With independent sensors, the HMM-based system again proved to be best.

6. Contributions and Future Research

This chapter describes research contributions and suggests future research.

6.1 Research contributions

Contributions from this dissertation research are in the following areas:

- Development of an HMM-based time series classifier
- Extension of Laine's CID optimization framework to include out-of-library performance
- Development of an out-of-library classification methodology
- Development of a target pose-estimation methodology using principal component analysis
- Application of the extended framework to a multi-class ATR experiment that competes the HMM-based classifier against a template-based classifier
- Development of the framework to allow classifiers to make reject, or not declare, decisions, to test classifiers against out-of-library records, and to measure the performance of three different fusion methods
- Development of evidence for independent optimal threshold settings for label fusion

6.1.1 Literature review

A comprehensive review of the literature covers the theory and development of hidden Markov models. The application of HMMs to ATR problems using high range-resolution radar signatures as features is described in Sec. 2.1.3.10, and it reveals limitations in treatment of prior knowledge of target aspect, inclusion of a

rejection option, and performance considering out-of-library targets. Other research areas covered in the literature review include model complexity in HMMs, multiple classifier fusion, rejection theory, and Laine's CID optimization framework.

6.1.2 Development of HMM-based classifier

Chapter 3 describes the development of an HMM-based time series classifier. Ultimately, the methodology results in a multi-dimensional Gaussian HMM operating on HRR-derived feature data. The model takes as input a sequence of feature data ordered by target aspect angle. The model develops relationships between the observation distribution associated with each hidden state and the signature of the target within a range of aspect angles.

6.1.3 Extended CID framework

Chapter 4 extends Laine's CID optimization framework by including an out-of-library performance measure. The framework retains the desired characteristic of allowing trade-off analysis without explicit classification error costs.

6.1.4 Development of out-of-library methodology

Section 4.3.2.5 describes a methodology whereby a classifier assigns an estimated posterior probability of out-of-library class membership to a test record. This methodology is implemented as a post-processing step after the classifier trained on in-library classes has adjudicated the test record. The methodology produces the estimated out-of-library posterior probability as a function of the in-library class posterior probabilities produced by the classifier. At some experimental settings the out-of-library discriminator correctly identified 60% of out-of-library records.

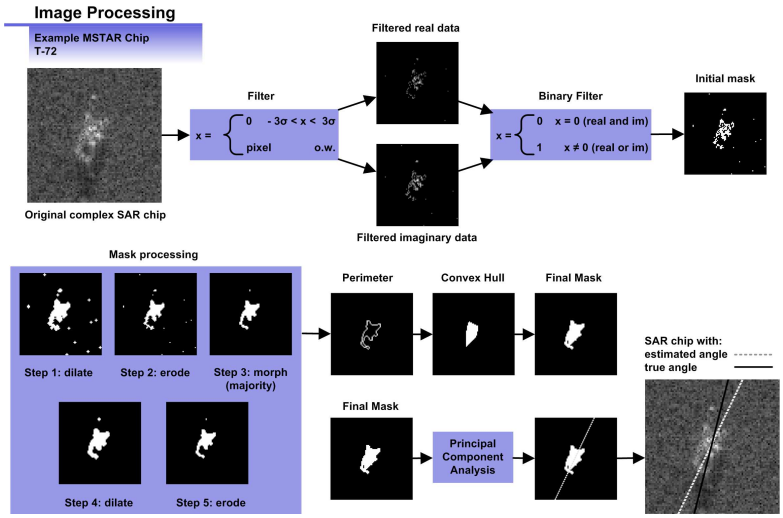

Figure 76. SAR chip image processing steps lead to a target mask that is evaluated using principal component analysis, resulting in an estimated target pose.

6.1.5 Development of target pose estimator

Section 5.4.5 develops an on-line method to estimate target aspect angle based on a target mask of a SAR image. The method uses principal component analysis to find the major axis of the target mask. An initial experiment found pose estimation error to be roughly 11°. Figure 76 highlights the steps taken to estimate target pose.

6.1.6 Application of extended CID framework

Chapter 5 details the application of the extended CID framework to an ATR experiment using DCS radar SAR data. The experiment competes an HMM-based system (a derivative of the Chapter 3 system) against a template-based classifier. The extended framework allows the systems to be compared inclusive of warfighter constraints, rejection option, and out-of-library target records. Results show that the HMM-based system provides the warfighter with better and more robust performance across a variety of experiment settings, including fusion rule, hostile/friend class

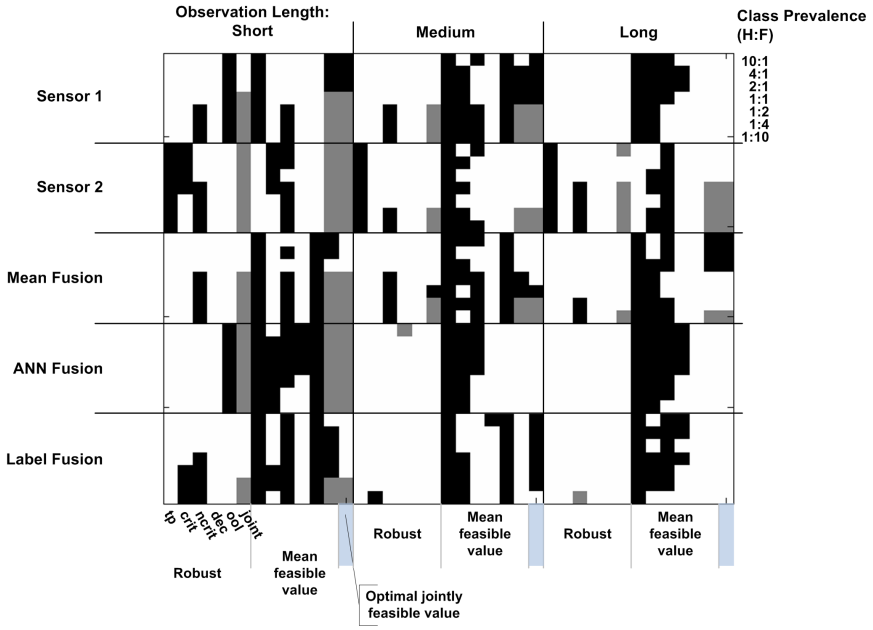

Figure 77. Differential results across experimental settings with co-located sensors and no prior target aspect knowledge. White portions indicate better performance for HMM-based classifier, black portions indicate better performance for template-based classifier, and gray indicates equal performance.

prevalence, observation length, and prior knowledge of target aspect angle. Figure 77 shows that the HMM-based classifier (white) is preferred over the template-based classifier (black) at a majority of experimental settings. Also, the size of feasible region in the threshold space is shown to provide a simple comparative measure of classifier robustness, and performance surfaces are shown to convey performance information and trade-space efficiently.

6.1.7 Evidence of independent threshold setting in fused system

Laine's research [15] demonstrates that independent thresholding for each classifier prior to applying label fusion allows improved performance over the application of a single threshold after the fusion of classifier outputs. Section 5.6.2.5 shows that independent thresholding yields optimal thresholds in different locations of the

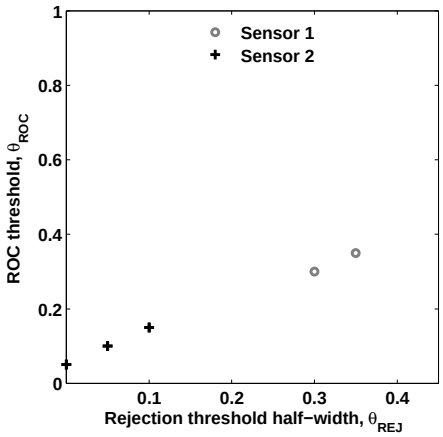

Figure 78. Optimal threshold settings for classifiers 1 and 2 when label fusion is applied. Note settings occur in different locations for each classifier.

threshold space for each classifier. This added flexibility allows the label fusion method to combine a classifier that performs well in one performance measure but poorly elsewhere with a second classifier whose threshold setting allows it to perform well in another area. Figure 78 shows an example of the optimal threshold settings for two classifiers.

6.2 Future research

Two areas for future research follow.

1. This research chooses an arbitrary feature set from HRR signatures of the targets. The literature does not identify a method for determining an appropriate number of features to extract from HRR profiles. The number of scattering centers is most likely related to target type and signal-to-noise ratio in the SAR image. Thus a feature saliency methodology could be developed to choose HRR features.

2. Converting the output of multiple HMM classifiers to class posterior probabilities allows the implementation of a Bayesian network that could learn a Bayes-optimal combination of classifiers with respect to classification accuracy.

This concept could also be implemented at the feature level to determine an optimal weighting of the selected features used in classification.

Appendix A. List of Abbreviations

ACC/DRSA	Air Combat Command's CID issues branch
AFDD	Air Force Doctrine Document
AFIT	Air Force Institute of Technology
AFPAM	Air Force pamphlet
AFRL	Air Force Research Laboratory
AIC	Akaike's information criterion
ANN	Artificial neural network
ATD/R	Automatic target detection and recognition
ATR	Automatic target recognition
BIC	Bayesian information criterion
CAD	Computer-aided design
CID	Combat identification
DAI-DAO	Data in – data out fusion
DARPA	Defense Advanced Research Projects Agency
DEI-DEO	Decision in – decision out fusion
DNA	Deoxyribonucleic acid
DTMC	Discrete time Markov chain
EM	Expectation maximization
EOC	Extended operating conditions
FEI-DEO	Features in – decision out fusion
FEI-FEO	Features in – features out fusion
FFT	Fast Fourier transform
FLIR	Forward-looking infrared
FN	Friend/neutral
FOS	Family of systems
HMM	Hidden Markov model
HRR	High range-resolution radar
ISR	Intelligence surveillance and reconnaissance
JFC	Joint Forces Commander
JP	Joint Publication

MCS	Multiple classifier system
MLE	Maximum likelihood estimator
MLPNN	Multi-layer perceptron neural network
MP	Mathematical programming
MSTAR	Moving and stationary target acquisition and recognition
OH	Other hostile
OOL	Out-of-library
PCA	Principal component analysis
PCS	Probability of correct selection
ROC	Receiver operating characteristic
ROI	Region of interest
SAR	Synthetic aperture radar
SNL	Sandia National Laboratory
TOD	Target of the day
TPR	True-positive rate
UCSC	University of California at Santa Cruz
USAF	United States Air Force

Appendix B. MATLAB code

B.1 run_script.m

```
% Tim Albrecht
% AFIT/ENS
% Sep 2005

% this script will perform multiple runs of the HMM DCS Project

clear
close all

% disable warning messages
warning off all

% add path to m-files from HMM Toolbox
addpath ([pwd, '\tools'])

%%%%%%%%%%%%%%%%%%%
% Data Structures %
%%%%%%%%%%%%%%%%%%%

ghmm_results = struct('log_lik',[]);
% structure to save log-likelihoods at each experiment setting
% 'ghmm_results(target,num_states,seq_length,feature_set).log_lik = ...
%      [test records, label class]'
ool_results = struct('log_lik',[]);
% out of library test results
% 'ool_results(target,num_states,seq_length,feature_set).log_lik = ...
%      [test records, label class]'
sensor_post = struct('post',[],'ool_post',[]);
% structure to save posteriors from each sensor (i.e. feature set)
% at each experiment setting
% 'sensor_post(target,num_states,seq_length,feature_set).post = ...
%      [test records, label class]'
loglik_post = struct('post',[],'ool_post',[]);
% structure to save posteriors from mean loglik fused sensors at
% each experiment setting
% 'loglik_post(data_class,num_states,seq_length).post = ...
%      [records label class]'
ann_post = struct('post',[],'ool_post',[]);
% structure to save posteriors from ann fused sensors at
% each experiment setting
% 'ann_post(num_states,seq_length).post = ...
%      [records label class]'
save([pwd,'\output'],'ghmm_results','ool_results','sensor_post',...
    'loglik_post','ann_post')

init_hmm = struct('prior',[],'trans',[],'mu',[],'sigma',[]);
```

```
% structure to save init ghmm parameters at each experiment setting
% 'init_hmm(target,num_states,feature_set).prior = []'
train_hmm = struct('prior',[],'trans',[],'mu',[],'sigma',[]);
% structure to save trained ghmm parameters at each experiment setting
% 'train_hmm(target,num_states,feature_set).prior = []'
ann = struct('net',[]);
% structure to save trained ann fusor
% 'ann(num_states,seq_length).net'
ann_data = struct('log_lik',[],'post',[]);
% data structure to hold training records, log-liks, and posteriors
% 'ann_data(target,feature_set,seq_length).log_lik = [record targetHMM]
%                                         .post = [record targetHMM]'
save([pwd,'\param'],'init_hmm','train_hmm','ann','ann_data');

%%%%%%%%%%%
% Settings %
%%%%%%%%%%%

settings = struct('num_states',[],'test_length',[],...
    'feature_sets',[],'feature_num',[],'targets',[]);
save([pwd,'\data'],'settings')
% '-.num_states' is the number of hidden states in the hidden
%     Markov model; defines the complexity of the model.
% '-.test_length' is the number of observation in a single test record;
%     here we have various settings, all less than or equal to the training
%     sequence length.
% '-.feature_sets' name (and number) of feature sets in experiment
% '-.feature_num' number of features in each feature set
% '-.targets' name (and number) of target classes in experiment

settings.num_states = [10 20 30 40 60 72 90];
settings.test_length = [4 10 20];
settings.feature_sets = {'HH','VV'};
settings.feature_num = [10 10];

% order the target list with 1 TOD, 4 OH, and 5 FN
settings.targets = {'target_1','target_2','target_5','target_10',...
                    'target_13','target_6','target_7','target_11',...
                    'target_12','target_15'};
save([pwd,'\data'],'settings','-append')

%%%%%%%%%%%%
% Scripting %
%%%%%%%%%%%%

for i = 1:size(settings.num_states,2)       % hidden state loop
    state = settings.num_states(i);

    % clocking mechanism
    tic
    disp(' ')
```

```
    disp(['begin train/test scripting at ',int2str(state),...
        ' states'])

    % 'build_train' builds training seq across 360 deg
    build_train

    % 'train_HMM' trains hmms for each target type and each feature set at
    % the given experiment settings
    train_HMM(state)

    t = toc;
    disp(['trained at ',int2str(state),' states. Time = ',...
        num2str(t/60),' mins'])

    for k = 1:size(settings.test_length,2)     % sequence length loop
        seq_length = settings.test_length(k);

        % clocking mechanism
        tic

        % 'build_trainANN' builds train records for ANN fusor training
        build_trainANN(state, seq_length)

        % 'train_ANN' trains ANN fuser using HMM outputs given sequences
        % from training set
        train_ANN(state, seq_length)

        % 'build_test' builds 100 test records of each target type for
        % each feature set. the initial aspect angle of the records is
        % chosen randomly and spans an interval of angles defined by
        % 'seq_length'.
        build_test(state, seq_length)

        % 'test_HMM' takes the test records and evaluates them using
        % the trained HMMs.
        test_HMM(state, seq_length)

        % fuse HMM outputs using trained ANNs
        test_ANN(state, seq_length)

        % test trained HMMs using out-of-library records, fuse using ANN
        % fuser, record results
        test_outlibrary(state, seq_length)

        t = toc;
        disp(['tested at sequence length ',int2str(seq_length),...
        '. Time = ',num2str(t/60),' mins'])
    end
end
```

B.2 build_train.m

```
function build_train

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'build_train' builds the training data sets for the HMMs

% load settings
load([pwd,'\data'],'settings')
% settings.feature_sets = {'HH','VV'};
% settings.targets = {'target_1','target_2','target_5','target_10',...
%                     'target_13','target_6','target_7','target_11',...
%                     'target_12','target_15'};

% create data structure to store training records
train_records = struct('data',[]);
% train_records(target,feature_set).data = [target exemplar];

% build training sequence across 360 degrees of aspect angle
for target = 1:length(settings.targets) % num targets

    for feature_set = 1:length(settings.feature_sets) % num feature sets

        % load the appropriate feature set/target data into structure
        % called 'data'
        load_str = [pwd,'\trial8_data\','train_',...
            settings.feature_sets{feature_set}];
        target_str = [settings.targets{target},'_',...
            settings.feature_sets{feature_set}];
        data = load(load_str,target_str);
        eval(['data = data.',settings.targets{target},'_',...
            settings.feature_sets{feature_set},';'])

        for num_exemplar = 1:1%size(data,2) for case of multiple exemplars
            temp_data = data(num_exemplar).feature;

            train_records(target,feature_set).data = temp_data;

        end % end exemplar loop
    end % end feature set loop
end % end target set loop

save([pwd,'\data'],'train_records','-append')
```

B.3 train_HMM.m

```
function train_HMM(num_states)

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'train_HMM' trains the HMMs used in the classification scheme.

% load the training data sequences and settings
load ([pwd,'\data'],'train_records','settings')
% 'train_records(target,feat_set).data = [feature_dim obs];'
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
% settings.targets = {'target_1','target_2','target_5','target_10',...
%                     'target_13','target_6','target_7','target_11',...
%                     'target_12','target_15'};

% load the ghmm parameter data structures
load ([pwd,'\param'],'init_hmm','train_hmm')

% set aspect window bins according to number of hidden states
binwidth = round((360 - 1) ./ num_states);
xx = binwidth*(0:num_states); xx(1) = 1;
xx(length(xx)) = 360;

% create initial HMM matricies: the prior, the state transition
% and mu and sigma of observation distributions
for target = 1:length(settings.targets) % num targets

    for feature_set = 1:length(settings.feature_sets) % num feature sets

            % force training sequence to begin in state 1
            init_hmm(target,num_states,feature_set).prior = ...
                zeros(num_states,1);
            init_hmm(target,num_states,feature_set).prior(1,1) = 1;

            % create bi-diagonal hidden state structure
            init_hmm(target,num_states,feature_set).trans = ...
                zeros(num_states);
            for i = 1:num_states
                fwd_state = mod(i, num_states) + 1;
                init_hmm(target,num_states,feature_set).trans(i,fwd_state) = 0.5;
                init_hmm(target,num_states,feature_set).trans(i,i) = 0.5;
            end

            temp = train_records(target,feature_set).data;

            % initialize mu and sigma by averageing and taking std of bin'd
            % feature observations (determined by number of states)
```

```
            for i = 1:num_states
                init_hmm(target,num_states,feature_set).mu(:,i) = ...
                    mean(temp(:,(xx(i):xx(i+1))),2);
                init_hmm(target,num_states,feature_set).sigma(:,:,i) =  ...
                    diag(std(temp(:,(xx(i):xx(i+1))),0,2));
            end
    end % end feature set loop
end % end target loop

% train the hmms, one per target per feature set
for target = 1:length(settings.targets) % num targets

    for feature_set = 1:length(settings.feature_sets) % num feature sets

        [LL, train_hmm(target,num_states,feature_set).prior,...
             train_hmm(target,num_states,feature_set).trans,...
             train_hmm(target,num_states,feature_set).mu,...
             train_hmm(target,num_states,feature_set).sigma,...
             mu_history] = ...
        learn_ghmm(num2cell(train_records(target,feature_set).data,[1 2]),...
            init_hmm(target,num_states,feature_set).prior,...
            init_hmm(target,num_states,feature_set).trans,...
            init_hmm(target,num_states,feature_set).mu,...
            init_hmm(target,num_states,feature_set).sigma,...
            'max_iter',10,'thresh',1e-5,'verbose',0,'cov_type','diag');

    end % end feature set loop
end % end target loop

save([pwd,'\param'],'init_hmm','train_hmm','-append')
```

B.4 build_trainANN.m

```
function build_trainANN(num_states, seq_length)

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'build_trainPNN' takes sequence length as input and builds the
% training seq to be used to train the neural network fuser in the
% classification scheme also uses num_states to determine initial state
% for testing sequences (prior aspect knowledge case)

% load settings
load([pwd,'\data'],'settings')
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
% settings.targets = {'target_1','target_2','target_5','target_10',...
%                     'target_13','target_6','target_7','target_11',...
%                     'target_12','target_15'};

train_records_ann = struct('record',[]);
% train_records_ann(target,feature_set,record#).record =
%           [feature_dim obs];

% create data structure to store test records
train_index = struct('prior',[]);
% train_index(target,state,seq_length,record).prior = column vector
% containing initial state distribution of each test record, revised to
% represent aspect knowledge to within +/- 22.5 deg independent of num of
% hidden states

% generate random starting aspect angles; create 200 start points per
% target type;
for target = 1:length(settings.targets) % num targets

    num_exemplar = 1;%size(data,2); multi exemplar case

    % pick random start points for ann training set
    index = randperm(360);
    ann_index = sort(index(1:100));%200));

    % insert code to consider prior target azimuth knowledge
    ang_cov = 360/num_states; % observation wedge covered by single state
    ang_cov_half = ang_cov/2; % half-width
    state_cov = round(22.5/ang_cov_half); % num states needed to cover 45 deg
    state_prob = 1/state_cov; % uniform prior over number of reqd states

    % builds a uniform distribution across the appropriate number of hidden
    % states within the prior dist vector
    for i = 1:size(ann_index,2)
```

```
    temp = zeros(num_states,1);
    est_state = floor((ann_index(i)-1)/ang_cov)+1;
    if rem(state_cov,2) ~= 0 %odd num states
        temp(est_state) = state_prob; % mid-point
        for j = 1:(state_cov-1)/2
            % lower half
            if est_state - j <= 0
                temp_index = mod(est_state - j + num_states,num_states + 1);
            else
                temp_index = est_state - j;
            end
            temp(temp_index) = state_prob;

            % upper half
            if est_state + j > num_states
                temp_index = mod(est_state + j,num_states);
            else
                temp_index = est_state + j;
            end
            temp(temp_index) = state_prob;
        end
    else    % even num states
        for j = 1:state_cov/2
            % lower mid-points
            if est_state - j + 1 <= 0
                temp_index = ...
                    mod(est_state-j+1+num_states,num_states+1);
            else
                temp_index = est_state - j + 1;
            end
            temp(temp_index) = state_prob;

            % upper mid-points
            if est_state + j > num_states
                temp_index = mod(est_state + j,num_states);
            else
                temp_index = est_state + j;
            end
            temp(temp_index) = state_prob;
        end
    end
    train_index(target,num_states,seq_length,i).prior = temp;
end

% build test sequences by target type (ann train set)
for i = 1:size(ann_index,2) % num test records per target type

    % for each record, must choose from among exemplars of target class
    tar_exemplar = randperm(num_exemplar);
    tar_exemplar = tar_exemplar(1);
```

```
        for feature_set = 1:length(settings.feature_sets) % num feature sets

            % load the appropriate feature set/target data into structure
            % called 'data'
            load_str = [pwd,'\trial8_data\','train_',...
                settings.feature_sets{feature_set}];
            target_str = [settings.targets{target},'_',...
                settings.feature_sets{feature_set}];
            data = load(load_str,target_str);
            eval(['data = data.',settings.targets{target},'_',...
                settings.feature_sets{feature_set},';'])
            % data = data(tar_exemplar);

            % pull full feature data into temp structure
            temp = data.feature;
            temp = [temp temp]; % account for possibility of wrapping around
                                % from 360 degrees back to 1
            % crop to seq_length
            temp2 = temp(:,ann_index(i):(ann_index(i) + seq_length-1));

            train_records_ann(target,feature_set,i).record = temp2;
        end % end feature set loop
    end % end ann train sequence loop
end % end target type loop

save ([pwd,'\data'],'train_records_ann','train_index','-append')
```

B.5 train_ANN.m

```
function train_ANN(num_states, seq_length)

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'train_ANN' feeds training records to the trained HMMs, produces
% log-likelihoods, converts to posterior probs, uses these posteriors to
% train an ANN fuser, and saves the trained ANN fusers.

% load the trained HMM parameters and ANN networks
load ([pwd,'\param'],'train_hmm','init_hmm','ann','ann_data')
% 'train_hmm(target,num_states,feature_set).prior = []'
% 'apnn(num_states,seq_length).net'
% 'ann_data(target,feature_set,seq_length).log_lik = [record targetHMM]
%                                          .post = [record targetHMM]'

% load the training sequences
load ([pwd,'\data'],'train_records_ann','train_index','settings')
% 'train_records_ann(target,feature_set,record#).record =
%       [feature_dim observation]'
% 'train_index(target,state,seq_length,record).prior = column vector'
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
% settings.targets = {'target_1','target_2','target_5','target_10',...
%                     'target_13','target_6','target_7','target_11',...
%                     'target_12','target_15'};

% send training records to trained HMMs; produce log-likelihoods
for target = 1:length(settings.targets) % class 'target' data

    for j = 1:length(settings.targets) % against class 'j' trained hmms

        for feature_set = 1:length(settings.feature_sets) % num feature sets

            ghmm_ll = [];

            for k = 1:size(train_records_pnn,3) % num test records

                % insert code to force prior aspect knowledge
               prior = ...
                   train_index(target,num_states,seq_length,k).prior;

                % for no knowledge use
%                 prior = normalise(ones(num_states,1));

                ghmm_ll(k) = ...
                    log_lik_ghmm(train_records_ann(target,feature_set,k).record,...
                    prior, ...
```

```
                    train_hmm(j,num_states,feature_set).trans, ...
                    train_hmm(j,num_states,feature_set).mu,...
                    train_hmm(j,num_states,feature_set).sigma);

            end % test record loop

            ghmm_ll = ghmm_ll';
            ann_data(target,feature_set,seq_length).log_lik(:,j) = ghmm_ll;
        end % end feature set loop
    end % end against j trained hmm loop
end % end data target type loop

% add truth to log-liks, build training set for ANN
data = [];
for feature_set = 1:size(ann_data,2) % num feature sets

    temp_holder = [];

    for target = 1:size(ann_data,1) % num targets
        temp_holder = ...
            [temp_holder ann_data(target,feature_set,seq_length).log_lik'];
    end
    % data ends up having (num_targets)*num_featuresets rows and
    % num_targets*num_records columns
    data = [data; temp_holder];
end % end feature set loop

% preprocess data to -1 1 range, save min/max for preprocessing testing
% data
[data min_d max_d] = premnmx(data);

num_records = size(train_records_ann,3);

truth = zeros(size(ann_data,1), size(ann_data,1)*num_records);
for i = 1:size(ann_data,1)
    truth(i,((i-1)*num_records+1):(i*num_records)) = ones(1,num_records);
end

% build FFMLP net
net = newff([-1*ones(20,1) ones(20,1)], [40 10],...
    {'tansig','logsig'});

% set parameters
net.trainFcn = 'traingdx';
net.trainParam.epochs = 3000;
net.trainParam.show = NaN;
net.trainParam.goal = .00001;

% train net
[net tr] = train(net,data,truth);
```

```
% save net
ann(num_states,seq_length).net = net;

% save trained ANN fuser
save([pwd,'\param'],'ann','min_d','max_d','ann_data','-append')
```

B.6 build_test.m

```
function build_test(num_states, seq_length)

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'build_test' takes sequence length as input and builds the
% testing data set to be used in the classification experiment
% also uses num_states to determine initial state for testing sequences
% (prior aspect knowledge case)

% load settings
load([pwd,'\data'],'settings')
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
% settings.targets = {'target_1','target_2','target_5','target_10',...
%                     'target_13','target_6','target_7','target_11',...
%                     'target_12','target_15'};

% create data structure to store test records
test_records = struct('record',[]);
% test_records(target,feature_set,record).record = [feature_dim observation];
test_index = struct('prior',[]);
% test_index(target,state,seq_length,record).prior = column vector
% containing prior distribution using prior aspect knowledge

% generate random starting aspect angles; create 100 start points per
% target type;
for target = 1:length(settings.targets) % num targets

    num_exemplar = 1;%size(data,2); multiple exemplar case

    % pick random start points for hmm_test set and pnn_train set
    index = randperm(360);
    hmm_test = sort(index(1:100));

    % insert code to consider prior target azimuth knowledge
    ang_cov = 360/num_states; % aspect angle wedge covered by single state
    ang_cov_half = ang_cov/2; % half-width
    state_cov = round(22.5/ang_cov_half); % num states needed to cover 45 deg
    state_prob = 1/state_cov; % uniform prior over number of reqd states

    % builds a uniform distribution across the appropriate number of hidden
    % states within the prior dist vector
    for i = 1:size(hmm_test,2)
        temp = zeros(num_states,1);
        est_state = floor((hmm_test(i)-1)/ang_cov)+1;
        if rem(state_cov,2) ~= 0 %odd num states
            temp(est_state) = state_prob; % mid-point
```

```
        for j = 1:(state_cov-1)/2
            % lower half
            if est_state - j <= 0
                temp_index = mod(est_state - j + num_states,num_states + 1);
            else
                temp_index = est_state - j;
            end
            temp(temp_index) = state_prob;

            % upper half
            if est_state + j > num_states
                temp_index = mod(est_state + j,num_states);
            else
                temp_index = est_state + j;
            end
            temp(temp_index) = state_prob;
        end
    else    % even num states
        for j = 1:state_cov/2
            % lower mid-points
            if est_state - j + 1 <= 0
                temp_index = ...
                    mod(est_state-j+1+num_states,num_states+1);
            else
                temp_index = est_state - j + 1;
            end
            temp(temp_index) = state_prob;

            % upper mid-points
            if est_state + j > num_states
                temp_index = mod(est_state + j,num_states);
            else
                temp_index = est_state + j;
            end
            temp(temp_index) = state_prob;
        end
    end
    test_index(target,num_states,seq_length,i).prior = temp;
end

% build test sequences by target type (hmm test set)
for i = 1:size(hmm_test,2) % num test records per target type

    % for each record, must choose from among exemplars of target class
    tar_exemplar = randperm(num_exemplar);
    tar_exemplar = tar_exemplar(1);

    for feature_set = 1:length(settings.feature_sets) % num feature sets

        % load the appropriate feature set/target data into structure
        % called 'data'
```

```
            load_str = [pwd,'\trial8_data\','test_',...
                settings.feature_sets{feature_set}];
            target_str = [settings.targets{target},'_',...
                settings.feature_sets{feature_set}];
            data = load(load_str,target_str);
            eval(['data = data.',settings.targets{target},'_',...
                settings.feature_sets{feature_set},';'])
            % data = data(tar_exemplar);

            % pull full feature data into temp structure
            temp = data.feature;
            temp = [temp temp]; % account for possibility of wrapping around
                                % from 360 degrees back to 1
            % crop to seq_length
            temp2 = temp(:,hmm_test(i):(hmm_test(i) + seq_length-1));

            test_records(target,feature_set,i).record = temp2;
        end % end feature set loop
    end % end hmm test sequence loop
end % end target type loop

save ([pwd,'\data'],'test_records','test_index','-append')
```

B.7 test_HMM.m

```
function test_HMM(num_states, seq_length)

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'test_HMM' evaluates the test records

% load the trained HMM parameters
load ([pwd,'\param'],'train_hmm','init_hmm')
% 'train_hmm(target,num_states,feature_set).prior = []'

% load the test sequences
load ([pwd,'\data'],'test_records','test_index','settings')
% 'test_records(target,feature_set,record#).record = [feature_dim obs]'
% 'test_index(target,state,seq_length).hmm = row vector
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
% settings.targets = {'target_1','target_2','target_5','target_10',...
%                     'target_13','target_6','target_7','target_11',...
%                     'target_12','target_15'};

% load the output information
load ([pwd,'\output'],'ghmm_results')
% 'ghmm_results(target,num_states,seq_length,feature_set).log_lik = ...
%      [test records, hmm class]'

for target = 1:length(settings.targets) % class 'target' data

    for j = 1:length(settings.targets) % against class 'j' trained hmms

        for feature_set = 1:length(settings.feature_sets) % num feature sets

            ghmm_ll = [];

            for k = 1:size(test_records,3) % num test records

                % insert code to use prior aspect angle information
              prior = ...
                  test_index(target,num_states,seq_length,k).prior;

                % for no knowledge use
%                 prior = normalise(ones(num_states,1));

                ghmm_ll(k) = ...
                 log_lik_ghmm(test_records(target,feature_set,k).record,...
                    prior, ...
                    train_hmm(j,num_states,feature_set).trans, ...
                    train_hmm(j,num_states,feature_set).mu,...
```

```
                    train_hmm(j,num_states,feature_set).sigma);

            end % test record loop

            ghmm_ll = ghmm_ll';
 ghmm_results(target,num_states,seq_length,feature_set).log_lik(:,j) = ...
     ghmm_ll;
        end % end feature set loop
    end % end against j trained hmm loop
end % end data target type loop

save([pwd,'\output'],'ghmm_results','-append')
```

B.8 test_ANN.m

```
function test_ANN(num_states, seq_length)

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'test_PNN' feeds log-likelihoods produced by the HMMs when given test
% sequences, feeds the log-liks to the trained ANN fusers.

% load the trained ANN networks and preprocessing values
load ([pwd,'\param'],'ann','min_d','max_d')
% 'ann(num_states,seq_length).net'

% load settings
load ([pwd,'\data'],'settings')
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
% settings.targets = {'target_1','target_2','target_5','target_10',...
%                     'target_13','target_6','target_7','target_11',...
%                     'target_12','target_15'};

% load the output of the HMM classifiers
load ([pwd,'\output'],'ghmm_results','ann_post')
% 'ghmm_results(target,num_states,seq_length,feature_set).log_lik = ...
%      [test records, label_class]'
% 'ann_post(num_states,seq_length).post = ...
%      [records, label_class]'

% build test set for ANN
data = [];
for feature_set = 1:size(ghmm_results,4) % num of feature sets

    temp_holder = [];

    for target = 1:size(ghmm_results,1) % num targets
        temp_holder = ...
         [temp_holder ...
          ghmm_results(target,num_states,seq_length,feature_set).log_lik'];
    end
    data = [data; temp_holder];
end % end feature set loop

% transform data to -1 1 range using min/max parameters
data = tramnmx(data,min_d,max_d);

temp = sim(ann(num_states,seq_length).net,data)';

% convert from (num_states,seq_length) which is matrix with
% num_rows=num_targets*num_records_per_target and num_cols=num_targets,
```

```
% to (data_class, num_states, seq_length) which is matrix with
% num_rows = num_records_per_target and num_cols = num_targets
for data_class = 1:length(settings.targets)   % index into data class
    m = ...
      size(ghmm_results(data_class,num_states,seq_length,1).log_lik,1);

    ann_post(data_class,num_states,seq_length).post = ...
        temp(((data_class-1)*m + 1):(data_class*m),:);
end

% save ANN fuser output
save([pwd,'\output'],'ann_post','-append')
```

B.9 test_outlibrary.m

```
function test_outlibrary(num_states,seq_length)

% Tim Albrecht
% AFIT/ENS
% Sep 2005

% 'test_outlibrary' performs out of library record testing. It builds a
% set of 100 test records drawn from 5 out of library targets (20 records
% each).

% load settings
load([pwd,'\data'],'settings')
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
target_list = {'target_3','target_4','target_8','target_9','target_14'};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generate test sequences from out of library targets %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% create data structure to store test records
ool_records = struct('record',[]);
% ool_records(target,feature_set,record).record = [feature_dim observation];
ool_index = struct('prior',[]);
% ool_index(target,state,seq_length,record).prior = column vector
% containing prior distribution using prior aspect knowledge

% generate random starting aspect angles; create 20 start points per
% target type;
for target = 1:length(target_list) % num targets

    % pick random start points
    index = randperm(360);
    ool_test = sort(index(1:20));

    % insert code to consider prior target azimuth knowledge
    ang_cov = 360/num_states; % aspect angle wedge covered by single state
    ang_cov_half = ang_cov/2; % half-width
    state_cov = round(22.5/ang_cov_half); % num states needed to cover 45 deg
    state_prob = 1/state_cov; % uniform prior over number of reqd states

    % builds a uniform distribution across the appropriate number of hidden
    % states within the prior dist vector
    for i = 1:size(ool_test,2)
        temp = zeros(num_states,1);
        est_state = floor((ool_test(i)-1)/ang_cov)+1;
        if rem(state_cov,2) ~= 0 %odd num states
            temp(est_state) = state_prob; % mid-point
            for j = 1:(state_cov-1)/2
```

```
            % lower half
            if est_state - j <= 0
                temp_index = ...
                    mod(est_state - j + num_states,num_states + 1);
            else
                temp_index = est_state - j;
            end
            temp(temp_index) = state_prob;

            % upper half
            if est_state + j > num_states
                temp_index = mod(est_state + j,num_states);
            else
                temp_index = est_state + j;
            end
            temp(temp_index) = state_prob;
        end
    else    % even num states
        for j = 1:state_cov/2
            % lower mid-points
            if est_state - j + 1 <= 0
                temp_index = ...
                    mod(est_state-j+1+num_states,num_states+1);
            else
                temp_index = est_state - j + 1;
            end
            temp(temp_index) = state_prob;

            % upper mid-points
            if est_state + j > num_states
                temp_index = mod(est_state + j,num_states);
            else
                temp_index = est_state + j;
            end
            temp(temp_index) = state_prob;
        end
    end
    ool_index(target,num_states,seq_length,i).prior = temp;
end

% build test sequences by target type
for i = 1:size(ool_test,2) % num test records per target type

    for feature_set = 1:length(settings.feature_sets) % num feature sets

        % load the appropriate feature set/target data into structure
        % called 'data'
        load_str = [pwd,'\trial8_data\','ool_test_',...
            settings.feature_sets{feature_set}];
        target_str = [target_list{target},'_',...
            settings.feature_sets{feature_set}];
```

```
            data = load(load_str,target_str);
            eval(['data = data.',target_list{target},'_',...
                settings.feature_sets{feature_set},';'])

            % pull full feature data into temp structure
            temp = data.feature;
            temp = [temp temp]; % account for possibility of wrapping around
                                % from 360 degrees back to 1
            % crop to seq_length
            temp2 = temp(:,ool_test(i):(ool_test(i) + seq_length-1));

            ool_records(target,feature_set,i).record = temp2;
        end % end feature set loop
    end % end ool test sequence loop
end % end target type loop

save ([pwd,'\data'],'ool_records','ool_index','-append')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% test out of library sequences against in-library trained HMMs %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% load the trained HMM parameters
load ([pwd,'\param'],'train_hmm','init_hmm')
% 'train_hmm(target,num_states,feature_set).prior = []'

% load the test sequences
load ([pwd,'\data'],'ool_records','ool_index','settings')
% 'ool_records(target,feature_set,record#).record = [feature_dim obs]'
% 'ool_index(target,state,seq_length).prior = row vector
% settings.feature_sets = {'HH','VV'};
% settings.feature_num = [10 10];
target_list = {'target_3','target_4','target_8','target_9','target_14'};

% load the output information
load ([pwd,'\output'],'ool_results')
% 'ool_results(target,num_states,seq_length,feature_set).log_lik = ...
%      [test records, hmm class]'

for target = 1:length(target_list) % class 'ool target' data

    for j = 1:length(settings.targets) % against class 'j' trained hmms

        for feature_set = 1:length(settings.feature_sets) % num feature sets

            ghmm_ll = [];

            for k = 1:size(ool_records,3) % num test records

                % insert code to use prior aspect angle information
                prior = ...
```

```
                  ool_index(target,num_states,seq_length,k).prior;

                % for no knowledge use
%                 prior = normalise(ones(num_states,1));

                ghmm_ll(k) = ...
                    log_lik_ghmm(ool_records(target,feature_set,k).record,...
                    prior, ...
                    train_hmm(j,num_states,feature_set).trans, ...
                    train_hmm(j,num_states,feature_set).mu,...
                    train_hmm(j,num_states,feature_set).sigma);

            end % test record loop

            ghmm_ll = ghmm_ll';
ool_results(target,num_states,seq_length,feature_set).log_lik(:,j) = ...
    ghmm_ll;
        end % end feature set loop
    end % end against j trained hmm loop
end % end data target type loop

save([pwd,'\output'],'ool_results','-append')

% fuse with trained ANN networks

% load the trained ANN networks
load ([pwd,'\param'],'ann','min_d','max_d')
% 'ann(num_states,seq_length).net'

% load fused ann posteriors
load([pwd,'\output'],'ann_post')

% build test set for ANN
data = [];
for feature_set = 1:size(ool_results,4) % num of feature sets

    temp_holder = [];

    for target = 1:size(ool_results,1) % num targets
        temp_holder = ...
          [temp_holder ...
           ool_results(target,num_states,seq_length,feature_set).log_lik'];
    end
    data = [data; temp_holder];
end % end feature set loop

% transform data to -1 1 range using min/max parameters
data = tramnmx(data,min_d,max_d);

temp = ...
    sim(ann(num_states,seq_length).net,data)';
```

```
% convert from (num_states,seq_length) which is matrix with
% num_rows=num_targets*num_records_per_target and num_cols=num_targets,
% to (data_class, num_states, seq_length) which is matrix with
% num_rows = num_records_per_target and num_cols = num_targets
for data_class = 1:length(target_list)   % index into data class
    m = ...
      size(ool_results(data_class,num_states,seq_length,1).log_lik,1);

    ann_post(data_class,num_states,seq_length).ool_post = ...
        temp(((data_class-1)*m + 1):(data_class*m),:);
end
save([pwd,'\output'],'ann_post','-append')
```

References

1. S. Tzu, *The Art of War*, translated by Thomas Cleary, Shambhala Publications, Boston, MA, 1988.

2. Department of the Air Force, *Air Warfare, AFDD 2-1*, Washington DC: HQ USAF, 2000.

3. US Joint Forces Command, *Capstone Requirements Document: Combat Identification*, Norfolk VA: US JFCOM J-8, 2001.

4. C. Sadowski, "Combat identification: Where are we?." AFIT OPER 596 Guest Lecture, 2005.

5. Director, J-5, Joint Staff, *Joint Vision 2020*, Washington DC: JCS, 2000.

6. Director, J-7, Joint Staff, *Joint Warfare of the Armed Forces of the United States, JP 1*, Washington DC: JCS, 2000.

7. Department of the Air Force, *Air Force Basic Doctrine, AFDD 1*, Washington DC: HQ USAF, 2003.

8. Department of the Air Force, *Intelligence, Surveillance, and Reconnaissance Operations, AFDD 2-5.2*, Washington DC: HQ USAF, 1999.

9. Department of the Air Force, *USAF Intelligence Targeting Guide, AFPAM 14-210*, Washington DC: HQ USAF, 1998.

10. F. Roli, "Fusion of multiple pattern classifiers," in *8th National Conference of the Italian Association of Artificial Intelligence*, 2003.

11. G. Fumera, F. Roli, and G. Giacinto, "Reject option with multiple thresholds," *Pattern Recognition* **33**, pp. 2099–2101, 2000.

12. G. Fumera, F. Roli, and G. Vernazza, "A method for error rejection in multiple classifier systems," in *Proceedings of the 11th International Conf on Image Analysis and Processing*, pp. 454–458, 2001.

13. G. Fumera, I. Pillai, and F. Roli, "A two-state classifier with reject option for text categorisation," in *5th Int. Workshop on Statistical Techniques in Pattern Recognition (SPR 2004)*, **3138**, pp. 771–779, 2004.

14. G. Fumera and F. Roli, "Analysis of error-reject trade-off in linearly combined multiple classifiers," *Pattern Recognition* **37**(6), pp. 1245–1265, 2004.

15. T. Laine, *Optimization of Automatic Target Recognition with a Reject Option Using Fusion and Correlated Sensor Data*. PhD dissertation, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2005.

16. S. Pribyl, "Operations research approaches to capabilities analysis for combat identification," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2005.

17. K. Fielding, *Spatio-Temporal Pattern Recognition using Hidden Markov Models*. PhD dissertation, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1994.

18. L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE* **77**, pp. 257–285, 1989.

19. Z. Ghahramani, "An introduction to hidden markov models and bayesian networks," *International Journal of Pattern Recognition and Artificial Intelligence* **15**, pp. 9–42, 2001.

20. R. Duda, P. Hart, and D. Stork, *Pattern Classification*, John Wiley and Sons, New York, NY, second ed., 2001.

21. R. Elliot, L. Aggoun, and J. Moore, *Hidden Markov Models: Estimation and Control*, Springer, New York, NY, second ed., 1997.

22. I. MacDonald and W. Zucchini, *Hidden Markov and Other Models for Discrete-valued Time Series*, Chapman and Hall/CRC, Boca Raton FL, first ed., 1997.

23. L. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *Annals of Mathematical Statistics* **37**, pp. 1554–1563, 1966.

24. L. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *Annals of Mathematical Statistics* **41**(1), pp. 164–171, 1970.

25. A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society* **Series B 39**, pp. 1–38, 1977.

26. P. Bickel, Y. Ritov, and T. Ryden, "Asymptotic normality of the maximum-likelihood estimator for general hidden markov models," *Annals of Statistics* **26**(4), pp. 1614–1635, 1998.

27. B. Leroux, "Maximum-likelihood estimation for hidden markov models," *Stochastic Processes and their Applications* **40**, pp. 127–143, 1992.

28. Y. Ephraim and N. Merhav, "Hidden markov processes," *IEEE Transactions on Information Theory* **48**(6), pp. 1518–1569, 2002.

29. G. Rigoll and D. Willett, "A NN/HMM hybrid for continuous speech recognition with a discriminant nonlinear feature extraction," in *Advances in Neural Information Processing Systems*, 1997.

30. E. Trentin and M. Gori, "A survey of hybrid ANN/HMM models for automatic speech recognition," *Neurocomputing* **37**, pp. 91–126, 2001.

31. J. Bilmes, "Natural statistical models for automatic speech recognition," Tech. Rep. TR-99-016, International Computer Science Institute, 1999.

32. D. Kulp, D. Haussler, M. Reese, and F. Eeckman, "A generalized hidden markov model for the recognition of human genes in DNA," in *Proceedings of the Conference on Intelligent Systems in Molecular Biology, AAAI Press*, 1996.

33. S. Eddy, "Profile hidden markov models," *Bioinformatics Review* **14**(9), pp. 755–763, 1998.

34. V. Alexandrov and M. Gerstein, "Using 3d hidden markov models that explicitly represent spatial coordinates to model and compare protein structures," *BMC Bioinformatics* **5**(2), 2004.

35. N. Arica and F. Yarman-Vural, "One-dimensional representation of two-dimensional information for hmm based handwriting recognition," *Pattern Recognition Letters* **21**, pp. 583–592, 2000.

36. J. Cai and Z.-Q. Liu, "Hidden markov models with spectral features for 2d shape recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(12), pp. 1454–1458, 2001.

37. T. Hu, L. D. Silva, and K. Sengupta, "A hybrid approach of NN and HMM for facial emotion classification," *Patten Recognition Letters* **22**, pp. 1303–1310, 2002.

38. V. Krishnamurthy, J. Moore, and S.-H. Chung, "Hidden markov model signal processing in presence of unknown deterministic interferences," *IEEE Transactions on Automatic Control* **38**(1), pp. 146–152, 1993.

39. P. Gader, "Hidden markov models for sensor fusion of EMI and GPR," Tech. Rep. DTIC ADA-422405, University of Missouri, Columbia, 2003.

40. J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," Tech. Rep. TR-97-021, U.C. Berkeley, Department of Electrical Enginnering and Computer Science, 1998.

41. V. Kulkarni, *Modeling and Analysis of Stochastic Systems*, Chapman and Hall, London, first ed., 1995.

42. M. DeWitt, "High range resolution radar target identification using the prony model and hidden markov models," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1992.

43. A. MacDonald, "Classification of high range resolution radar returns using hidden markov and gaussian mixture models," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1999.

44. P. Runkle, P. Bharadwaj, L. Couchman, and L. Carin, "Hidden markov models for multiaspect target classification," *IEEE Transactions on Signal Processing* **47**(7), pp. 2035–2040, 1999.

45. P. Runkle, L. Carin, L. Couchman, T. Yoder, and J. Bucaro, "Multiaspect target identification with wave-based matched pursuits and continuous hidden markov models," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(12), pp. 1371–1378, 1999.

46. P. Bharadwaj, P. Runkle, L. Carin, J. Berrie, and J. Hughes, "Multi-aspect classification of airborne targets via physics-based hmms and mathcing pursuits," *IEEE Transactions on Aerospace and Electronic Systems* **37**(2), pp. 595–606, 2001.

47. X. Liao, P. Runkle, Y. Jiao, and L. Carin, "Identification of ground targets from sequential hrr radar signatures," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2897–2900, 2001.

48. X. Liao, P. Runkle, and L. Carin, "Identification of ground targets from sequential high-range-resolution radar signatures," *IEEE Transactions on Aerospace and Electronic Systems* **38**(4), pp. 1230–1242, 2002.

49. J. Li and P. Stoica, "Angle and waveform estimation via relax," *IEEE Transactions on Aerospace and Electronic Systems* **33**(3), pp. 1077–1087, 1997.

50. A. Paul and A. Shaw, "Robust HRR radar target identification by hybridization of HMM and eigen-template based matched filtering," in *Proceedings of SPIE*, **5094**, pp. 278–289, 2003.

51. D. Kottke, P. Fiore, K. Brown, and J.-K. Fwu, "A design for HMM-based SAR ATR," in *Proceedings of SPIE*, **3370**, pp. 541–551, 1998.

52. C. Nilubol, Q. Pham, R. Mersereau, M. Smith, and M. Clements, "Translational and rotational invariant hidden markov model for automatic target recognition," in *Proceedings of SPIE*, **3374**, pp. 179–185, 1998.

53. C. Nilubol, R. Mersereau, and M. Smith, "An improved hidden markov model classifier for SAR images," in *Proceedings of SPIE*, **3720**, pp. 113–122, 1999.

54. S. Jacobs and J. O'Sullivan, "Automatic target recognition using sequences of high resolution radar range-profiles," *IEEE Transactions on Aerospace and Electronic Systems* **36**(2), pp. 364–382, 2000.

55. D. Zhou, G. Liu, and J. Wang, "Spatio-temporal target identification method of high-range resolution radar," *Pattern Recognition* **33**, pp. 1–7, 2000.

56. B. Pei and Z. Bao, "Radar target recognition based on peak location of HRR profile and HMM classifiers," in *Proceedings of IEE Radar 2002 Conference*, pp. 414–418, 2002.

57. R. Williams, D. Gross, A. Palomino, J. Westerkamp, and D. Wardell, "1d HRR data analysis and ATR assessment," in *Proceedings of SPIE*, **3370**, pp. 588–599, 1998.

58. R. Williams, J. Westerkamp, D. Gross, A. Palomino, T. Kaufman, and T. Fister, "Analysis of a 1-d HRR moving target ATR," in *Proceedings of SPIE*, **3721**, pp. 413–424, 1999.

59. R. Williams, J. Westerkamp, D. Gross, and A. Palomino, "Automatic target recognition of time critical moving targets using 1d high range resolution radar," *IEEE AES Systems Magazine* **April**, pp. 37–43, 2000.

60. G. Meyer, *Classification of Radar Targets Using Invariant Features*. PhD dissertation, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2003.

61. M. Zumwalt, "Robust high range resolution radar for target classification," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2000.

62. R. Mitchell and J. Westerkamp, "Robust statistical feature based aircraft identification," *IEEE Transactions on Aerospace and Electronic Systems* **35**(3), pp. 1077–1094, 1999.

63. A. Shaw, R. Vashist, and R. Williams, "HRR-ATR using eigen-templates with noisy observations in unknown target scenario," in *Proceedings of SPIE*, **4053**, 2000.

64. T. Zajic, C. Rago, R. Mahler, M. Huff, and M. Noviskey, "Joint tracking, pose estimation and target recognition using HRRR and track data: New results," in *Proceedings of SPIE*, **4380**, pp. 196–206, 2001.

65. Sensor Data Management System, *MSTAR Database*, Air Force Research Laboratory, Wright-Patterson AFB, OH. `http://www.mbvlab.wpafb.af.mil/public/sdms/additional.htm`, 1995.

66. I. J. Myung, "The importance of complexity in model selection," *Journal of Mathematical Psychology* **44**, pp. 190–204, 2000.

67. M. Forster, "Key concepts in model selection: Performance and generalizability," *Journal of Mathematical Psychology* **44**, pp. 205–231, 2000.

68. A. Lanterman, "Schwarz, wallace, and rissanen: Intertwining themes in theories of model selection," *International Statistical Review* **69**(2), pp. 185–212, 2001.

69. K. P. Burnham and D. R. Anderson, *Model Selection and Multimodel Inference*, Springer, NY, second ed., 2002.

70. D. Li, A. Biem, and J. Subrahmonia, "HMM topology optimization for handwriting recognition," *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)* , pp. 1521–1524, 2001.

71. T. Ryden, "Estimating the order of hidden markov models," *Statistics* **26**, pp. 345–354, 1995.

72. R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions* **222**, pp. 309–368, 1922.

73. D. D. Wackerly, W. M. III, and R. L. Scheaffer, *Mathematical Statistics with Applications*, Duxbury Press, Belmont, CA, fifth ed., 1996.

74. A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, McGraw Hill, NY, third ed., 1974.

75. R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics*, Prentice Hall, NJ, fifth ed., 1995.

76. P. J. Bickel and K. A. Doksum, *Mathematical Statistics*, Holden-Day, CA, 1977.

77. S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics* **22**, pp. 79–86, 1951.

78. H. Akaike, "Information theory as an extension of the maximum likelihood principle," in *Second International Symposium on Information Theory*, B. Petrov and F. Csaki, eds., pp. 267–281, 1973.

79. G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics* **6**, pp. 2:461–464, 1978.

80. F. Roli and G. Giacinto, *Hybrid Methods in Pattern Recognition by H. Bunke and A. Kandel*, ch. 8: Design of Multiple Classifier Systems. World Scientific Publishing, 2002.

81. B. Dasarathy, *Multi-Sensor, Multi-Source Information Fusion: Architectures, Algorithms, and Applications*. International Society for Optical Engineering, Defense and Security Symposium Short Course, SC149, 2004.

82. L. Chan, S. Der, and N. Nasrabadi, "Dualband FLIR fusion for automatic target recognition," *Information Fusion* **4**, pp. 35–45, 2003.

83. S. Rizvi and N. Nasrabadi, "Fusion of FLIR automatic target recognition algorithms," *Information Fusion* **4**, pp. 247–258, 2003.

84. X. Song, Y. Abu-Mostafa, J. Sill, H. Kasdan, and M. Pavel, "Robust image recognition by fusion of contextual information," *Information Fusion* **3**, pp. 277–287, 2002.

85. S. Storm, "An investigation of the effects of correlation in sensor fusion," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2003.

86. N. Leap, "An investigation of the effects of correlation, autocorrelation, and sample size in classifier fusion," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2004.

87. P. Clemans, "An investigation of the optimal sensor ensemble for sensor fusion," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2004.

88. F. Mindrup, "An investigation of the effects of correlation and autocorrelation in classifier fusion with non-declarations," Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2005.

89. C. Chow, "On optimum rejection error and reject tradeoff," *IEEE Transactions on Information Theory* **IT-16**, pp. 41–46, 1970.

90. P. Devijver and J. Kittler, *Pattern Recognition – a Statistical Approach*, Prentice-Hall Int'l, London, 1982.

91. K. Haspert, "Optimum ID sensor fusion for multiple target types," Tech. Rep. D-2451, Institute for Defense Analysis (IDA), 2000.

92. T. Albrecht, "Hidden markov models for classifying genetic sequences." AFIT EENG 621 Pattern Recognition II, 2003.

93. K. Murphy, *Hidden Markov Model Toolbox for Matlab*. Massachusetts Institute of Technology, `http://www.ai.mit.edu/~murphyk/Software/HMM/hmm.html`, 2003.

94. R. Durbin, *Biological Sequence Analysis: probabalistic models of proteins and nucleic acids*, Cambridge University Press, Cambridge, UK, 1998.

95. University of California at Santa Cruz, Computational Biology, `http://www.soe.ucsc.edu/research/compbio`, 2003.

96. J. Swets, R. Dawes, and J. Monahan, "Better decisions through science," *Scientific American* **283**, pp. 82–87, 2000.

97. S. Alsing, *The Evaluation of Competing Classifiers*. PhD dissertation, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2000.

98. T. Ross, J. Bradley, L. Hudson, and M. O'Connor, "SAR ATR - so what's the problem? - an MSTAR perspective," in *Proceedings of SPIE*, **3721**, pp. 662–672, 1999.

99. T. Albrecht and S. Gustafson, "Hidden markov models for classifying SAR target images," in *Proceedings of SPIE*, **5427**, pp. 302–308, 2004.

100. T. Albrecht and K. Bauer, "Classification of sequenced SAR target images via hidden markov models with decision fusion," in *Proceedings of SPIE*, **5808**, pp. 306–313, 2005.

101. J. Principe, D. Xu, and J. F. III, "Pose estimation in SAR using an information theoretic criterion," in *Proceedings of SPIE*, **3370**, pp. 218–229, 1998.

102. L. Voicu, R. Patton, and H. Myler, "Multi-criterion vehicle pose estimation for SAR-ATR," in *Proceedings of SPIE*, **3721**, pp. 497–506, 1999.

103. Q. Zhao, J. Principe, V. Brennan, D. Xu, and Z. Wang, "Synthetic aperture radar automatic target recognition with three strategies of learning and representation," *Optical Engineering* **39**, pp. 1230–1244, 2000.

104. L. Kaplan and R. Murenzi, "Pose estimation of SAR imagery using the two dimensional continuous waelet transform," *Pattern Recognition Letters* **24**, pp. 2269–2280, 2003.

105. Y. Sun, Z. Liu, S. Todorovic, and J. Li, "Synthetic aperture radar automatic target recognition using adaptive boosting," in *Proceedings of SPIE*, **5808**, pp. 282–293, 2005.

Vita

Major Tim Albrecht was born in September 1971 in Bryan, Texas. He attended high school in Papillion, Nebraska and graduated in 1989 from O'Fallon Township High School in O'Fallon, Illinois. He studied electrical engineering at Northwestern University in Evanston, Illinois graduating in June 1993 with a bachelor of science degree. He was commissioned into the US Air Force through ROTC in June 1993 with subsequent assignments to the National Air and Space Intelligence Center, Wright-Patterson AFB, Ohio and the Air Force Studies and Analyses Agency, Pentagon, Washington DC. Maj Albrecht earned a master of science degree in operations research from the Air Force Institute of Technology in 1999 and is currently a Ph.D. candidate at AFIT in operations research. He will be assigned to the Air Force Logistics Management Agency, Maxwell AFB, Alabama upon completion of his program. He is a member of MORS, INFORMS, Omega Rho, Tau Beta Pi, and SPIE.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
15–09–2005	Doctoral Dissertation	Oct 2002 – Sep 2005

4. TITLE AND SUBTITLE

COMBAT IDENTIFICATION WITH SEQUENTIAL OBSERVATIONS, REJECTION OPTION, AND OUT-OF-LIBRARY TARGETS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Albrecht, Timothy W., Major, USAF

5d. PROJECT NUMBER

AFOSR grant #NMIPR045203616 ACC/DRSA

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way, Building 641,
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/DS/ENS/05-03

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFOSR/NM ATTN: Maj Todd Combs
Stuite 325 Rm 3112
875 Randolph St
Arlington VA 22203-1768

ACC/DRSA ATTN: Charles Sadowski
216 Hunting Ave, Rm 105
Langley AFB VA 23665-2777

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR/NM, ACC/DRSA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT This research extends a mathematical framework to select the optimal sensor ensemble and fusion method across multiple decision thresholds subject to warfighter constraints for a combat identification (CID) system. The formulation includes treatment of exemplars from target classes on which the CID system classifiers are not trained (out-of-library classes) and enables the warfighter to optimize a CID system without explicit enumeration of classifier error costs. A time-series classifier design methodology is developed and applied, yielding a multi-variate Gaussian hidden Markov model (HMM). The extended CID framework is used to compete the HMM-based CID system against a template-based CID system. The framework evaluates competing classifier systems that have multiple fusion methods, varied prior probabilities of targets and non-targets, varied correlation between multiple sensor looks, and varied levels of target pose estimation error. Assessment using the extended framework reveals larger feasible operating regions for the HMM-based classifier across experimental settings. In some cases the HMM-based classifier yields a feasible region that is 25% of the threshold operating space versus 1% for the template-based classifier.

15. SUBJECT TERMS

Combat identification; hidden Markov models; high range-resolution radar

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth W. Bauer, Jr., (AFIT/ENS)
U	U	U	UU	245	19b. TELEPHONE NUMBER *(include area code)* (937) 255–6565, ext 4328